EPFL

Thèse n° 10 541

Understanding generalization and robustness in modern deep learning

Présentée le 24 mai 2024

Faculté informatique et communications
Laboratoire de théorie en apprentissage automatique
Programme doctoral en informatique et communications

pour l'obtention du grade de Docteur ès Sciences

par

Maksym ANDRIUSHCHENKO

Acceptée sur proposition du jury

Prof. R. Guerraoui, président du jury
Prof. N. H. B. Flammarion, directeur de thèse
Prof. Z. Kolter, rapporteur
Dr S. Bubeck, rapporteur
Prof. F. Krzakala, rapporteur

École polytechnique fédérale de Lausanne

2024

Acknowledgements

First of all, I would like to express my deepest gratitude to my Ph.D. advisor, Nicolas Flammarion, for his guidance, support, and encouragement throughout my five-year long Ph.D. journey. His enthusiasm, energy, and readiness to explore new topics together with me have been fundamental to this thesis. Nicolas not only allowed me to have the freedom to pursue my own research interests but also provided valuable career advice, connected me with other people in the field, gave valuable feedback on many talks and write-ups, and otherwise was very helpful throughout the last five years. His support was absolutely essential in navigating my research transitions from adversarial robustness to generalization in deep learning, and then to understanding large language models and their robustness. Thank you so much, Nicolas, for being a great advisor and mentor.

I am also immensely grateful to my master's thesis advisor, Matthias Hein from the University of Tübingen, who essentially introduced me to this research field and inspired me to pursue a Ph.D. His incredible work ethic, research vision, and ability to produce impactful work—even with limited academic resources—have left a deep impression on me. Moreover, his support, advice, and collaboration have been very important to me also during my Ph.D.

My gratitude also extends to my Ph.D. committee members, Sebastien Bubeck, Zico Kolter, Florent Krzakala, and Rachid Guerraoui, for finding the time in their incredibly busy schedules and for asking very insightful questions during my defense. I appreciated both general questions about the field and my work, but also very specific ones, such as the one about the proof of one of the theorems presented in an appendix (I am still impressed by that question!).

I am fortunate to have collaborated with many talented people from whom I have learned really *a lot*. I owe a great deal to my frequent collaborators Francesco Croce, Marius Mosbach, Edoardo Debenedetti, and Aditya Varre. I am especially thankful to Francesco for many years of productive collaboration and for always providing an inspiring example of how to get things done. I am also very grateful to Marius for countless discussions about machine learning, natural language processing, and many other topics—it's a shame that we have written only two papers together! My appreciation also goes to my other collaborators: Maximilian Müller, Dara Bahri, Hossein Mobahi, Linara Adilova, Vikash Sehwag, Loucas Pillaud-Vivien, Klim Kireev, and Hao Zhao. Also, special thanks to my letter writers—Bernt Schiele, Dietrich Klakow, Prateek Mittal, and again, Nicolas Flammarion and Matthias Hein. I am also grateful to Eric Wong for his occasional advice

over the years which has been very valuable, and to John Collomosse, who hosted my internship at Adobe Research in Summer 2021.

I am indebted to the EPFL doctoral program, Google, and Open Philanthropy for funding my Ph.D., and to EPFL in general for providing an excellent academic environment. My thanks also go out to the vibrant machine learning community at large for countless insightful conversations with so many people, both in person at major conferences and online on platforms like Twitter.

I would like to also acknowledge my dear friends Arnout, Dongyang, Atli, Baran, Oğuzhan, Nastia, Thijs, Emma, Bogdan, Mariam, Guille, Tolis, Marius, Valentyn, and Rati. Without you, my Ph.D. journey would have been much less fun! Also, a special thanks to Bogdan for introducing me to EPFL back in Summer 2018. I must also thank my lab mates at TML and our neighbors at MLO—Scott, Maria, Aditya, Francescos, Oğuz, Gizem, Hristo, Felix, Alex, Keivan, Matteo, Praneeth, Tao, Jean-Baptiste, Sebastian, and Martin—who made the lab a very fun place to work and hang out. I am sorry if I forgot to mention someone here—there were *so* many amazing folks around!

Finally, I am incredibly grateful to my parents, Vadym and Olga, and my brother Igor for their unwavering support and for instilling in me a passion for math, programming, and *knowledge* from a very young age. I am incredibly fortunate to have such a supportive family. This thesis, of course, would be completely impossible without their constant support.

Lausanne, April 29, 2024 M. A.

Abstract

In this thesis, we study two closely related directions: *robustness* and *generalization* in modern deep learning. Deep learning models based on empirical risk minimization are known to be often non-robust to small, worst-case perturbations known as *adversarial examples* that can easily fool state-of-the-art deep neural networks into making wrong predictions. Their existence can be seen as a generalization problem: despite the impressive average-case performance, the deep learning models tend to learn non-robust features that can be used for adversarial manipulations. In this thesis, we delve deeply into a range of questions related to robustness and generalization, such as how to accurately evaluate robustness, how to make robust training more efficient, and why some optimization algorithms lead to better generalization and learn qualitatively different features.

We start the first direction from exploring computationally efficient methods to perform adversarial training and its failure mode referred to as *catastrophic overfitting* when the model suddenly loses its robustness after some point in training. Then we provide a better understanding of the robustness evaluation and the progress in the field by proposing new query-efficient black-box adversarial attacks based on random search that do not rely on the gradient information and thus can complement a typical robustness evaluation based on gradient-based methods. Finally, for the same goal, we propose a new community-driven robustness benchmark `RobustBench` which aims to systematically track the progress in the field in a standardized way.

We start the second direction from investigating reasons behind the success of sharpness-aware minimization, a recent algorithm that increases robustness in the parameter space during training and improves generalization for deep networks. Then we discuss why over-parameterized models trained with stochastic gradient descent tend to generalize surprisingly well even without any explicit regularization. We study the implicit regularization induced by stochastic gradient descent with large step sizes and its effect on the features learned by the model. Finally, we rigorously study the relationship between sharpness of minima (i.e., robustness in the parameter space) and generalization that prior works observed to correlate to each other. Our study suggests that, contrary to the common belief, sharpness is not a good indicator of generalization and it rather tends to correlate well with some hyperparameters like the learning rate but not inherently with generalization.

Keywords: Machine learning, deep learning, adversarial robustness, generalization, implicit regularization.

Résumé

Dans cette thèse, nous étudions deux directions étroitement liées : la *robustesse* et la *généralisation* dans l'apprentissage profond moderne. Les modèles d'apprentissage profond basés sur la minimisation du risque empirique sont connus pour être souvent non robustes aux petites perturbations dans le pire des cas, appelées *exemples adversariaux*, qui peuvent facilement tromper les réseaux de neurones profonds de pointe et les amener à faire de mauvaises prédictions. Leur existence peut être considérée comme un problème de généralisation : malgré des performances impressionnantes en moyenne, les modèles d'apprentissage profond ont tendance à apprendre des caractéristiques non robustes qui peuvent être utilisées pour des manipulations adversariales. Dans cette thèse, nous approfondissons une série de questions liées à la robustesse et à la généralisation, telles que comment évaluer précisément la robustesse, comment rendre l'entraînement robuste plus efficace et pourquoi certains algorithmes d'optimisation conduisent à une meilleure généralisation et apprennent des caractéristiques qualitativement différentes.

Nous commençons la première direction en explorant des méthodes computationnellement efficaces pour effectuer un entraînement adversarial et son mode d'échec appelé *surapprentissage catastrophique* lorsque le modèle perd soudainement sa robustesse après un certain point de l'entraînement. Ensuite, nous fournissons une meilleure compréhension de l'évaluation de la robustesse et des progrès dans le domaine en proposant de nouvelles attaques adversariales boite-noire efficaces en termes de requêtes, basées sur une recherche aléatoire qui ne repose pas sur les informations de gradient et peut donc compléter une évaluation typique de la robustesse basée sur des méthodes de gradient. Enfin, dans le même but, nous proposons un nouveau benchmark `RobustBench` qui vise à suivre systématiquement les progrès dans le domaine de la robustness de manière standardisée.

Nous commençons la deuxième direction en étudiant les raisons du succès de la minimisation sensible à la raideur, un algorithme récent qui augmente la robustesse dans l'espace des paramètres pendant l'entraînement et améliore la généralisation pour les réseaux profonds. Ensuite, nous discutons pourquoi les modèles surparamétrés entraînés avec une descente de gradient stochastique ont tendance à généraliser étonnamment bien même sans régularisation explicite. Nous étudions la régularisation implicite induite par la descente de gradient stochastique avec de grands pas et son effet sur les caractéristiques apprises par le modèle. Enfin, nous étudions rigoureusement la relation entre la raideur des minima (c'est-à-dire la robustesse dans l'espace des paramètres) et la généralisation que les travaux antérieurs ont observé être corrélés entre eux. Notre étude suggère que, en dépit des idées reçues, la raideur n'est pas un bon indicateur de généralisation et elle a plutôt ten-

dance à bien corréler avec certains hyperparamètres comme le taux d'apprentissage mais pas de manière inhérente avec la généralisation.
Mots-clés : Apprentissage automatique, apprentissage profond, robustesse adversariale, généralisation, régularisation implicite.

Contents

Acknowledgements i

Abstract (English/Français) iii

1 Introduction 1
- 1.1 Outline of the thesis . . . 4
- 1.2 Contributions beyond this thesis . . . 5

I Robustness in Modern Deep Learning 7

2 Understanding and Improving Fast Adversarial Training 9
- 2.1 Preface . . . 9
- 2.2 Introduction . . . 10
- 2.3 Problem overview and related work . . . 12
- 2.4 The role and limitations of using random initialization in FGSM training 14
- 2.5 Understanding catastrophic overfitting via gradient alignment . . . 16
- 2.6 Increasing gradient alignment improves fast adversarial training . . . 20
- 2.7 Conclusions and outlook . . . 22
- 2.8 Deferred proofs . . . 24
  - 2.8.1 Proof of Lemma 1 . . . 24
  - 2.8.2 Proof and discussion of Lemma 2 . . . 25
- 2.9 Experimental details . . . 29
- 2.10 Supporting experiments and visualizations for Sec. 2.4 and Sec. 2.5 . . . 31
  - 2.10.1 Quality of the linear approximation for ReLU networks . . . 31
  - 2.10.2 Catastrophic overfitting in a single-layer CNN . . . 32
- 2.11 Additional experiments for different adversarial training schemes . . . 35
  - 2.11.1 Stronger PGD-2 baseline . . . 35
  - 2.11.2 Results with early stopping . . . 35
  - 2.11.3 Results for specific ℓ_∞-radii . . . 36
  - 2.11.4 Ablation studies . . . 39
  - 2.11.5 Comparison of GradAlign to gradient-based penalties . . . 40

3 Square Attack: a Query-Efficient Black-Box Adversarial Attack via Random Search 45

3.1 Preface . . . 45
3.2 Introduction . . . 45
3.3 Related Work . . . 46
3.4 Square Attack . . . 48
3.4.1 Adversarial Examples in the l_p-threat Model . . . 48
3.4.2 General Algorithmic Scheme of the Square Attack . . . 49
3.4.3 The l_∞-Square Attack . . . 50
3.4.4 The l_2-Square Attack . . . 51
3.5 Theoretical and Empirical Justification of the Method . . . 53
3.5.1 Convergence Analysis of Random Search . . . 53
3.5.2 Why Squares? . . . 54
3.5.3 Ablation Study . . . 55
3.6 Experiments . . . 56
3.6.1 Evaluation on ImageNet . . . 57
3.6.2 Square Attack Can be More Accurate than White-box Attacks . . . 58
3.7 Conclusion . . . 60
3.8 Proofs Omitted from Section 3.4 and Section 3.5 . . . 61
3.8.1 Proof of Proposition 3.4.1 . . . 61
3.8.2 Proof of Proposition 3.5.1 . . . 61
3.8.3 Assumptions in Eq. (3.3) Do Not Hold for the Sampling Distribution P . . . 62
3.8.4 Assumptions in Eq. (3.3) Hold for the Sampling Distribution P^{multiple} 63
3.8.5 Why Updates of Equal Sign? . . . 64
3.8.6 Proof of Proposition 3.5.2 . . . 65
3.9 Experimental Details . . . 66
3.9.1 Experiments on ImageNet . . . 66
3.9.2 Square Attack Can be More Accurate than White-box Attacks . . . 67
3.10 Ablation Study . . . 67
3.10.1 l_∞-Square Attack . . . 68
3.10.2 l_2-Square Attack . . . 68
3.11 Stability of the Attack under Different Random Seeds . . . 69
3.12 Additional Experimental Results . . . 70
3.12.1 Targeted Attacks . . . 71
3.12.2 Success Rate on ImageNet for Different Number of Queries . . . 72
3.12.3 Performance on Architectures with Dilated Convolutions . . . 73
3.12.4 Imperceptible Adversarial Examples with the Square Attack . . . 74
3.12.5 Analysis of Adversarial Examples that Require More Queries . . . 74
3.12.6 Breaking the Post-averaging Defense . . . 77

4 `RobustBench`: a Standardized Adversarial Robustness Benchmark 79
4.1 Preface . . . 79
4.2 Introduction . . . 80
4.3 Background and related work . . . 81
4.4 Description of `RobustBench` . . . 84

4.4.1 Leaderboard . . . 84
4.4.2 Model Zoo . . . 88
4.5 Analysis . . . 89
4.6 Outlook . . . 93
4.7 Broader impact . . . 95
4.8 Licenses . . . 95
4.9 Maintenance plan . . . 96
4.10 Details of the ImageNet leaderboards . . . 97
4.11 Reproducibility and runtime . . . 97
4.12 Additional analysis . . . 98
4.13 Leaderboards . . . 104

II Generalization in Modern Deep Learning 111

5 Towards Understanding Sharpness-Aware Minimization 113
5.1 Preface . . . 113
5.2 Introduction . . . 114
5.3 Background on SAM . . . 115
5.4 Challenging the Existing Understanding of SAM . . . 117
5.5 Understanding the Generalization Benefits of SAM . . . 119
5.5.1 Testing Two Natural Hypotheses for Why Low m in m-SAM Could be Beneficial . . . 120
5.5.2 Provable Benefit of SAM for Diagonal Linear Networks . . . 121
5.5.3 Empirical Study of the Implicit Bias in Non-Linear Networks . . . 124
5.6 Understanding the Optimization Aspects of SAM . . . 126
5.6.1 Theoretical Analysis of Convergence of SAM . . . 127
5.6.2 Convergence of SAM for Deep Networks . . . 128
5.7 Conclusions . . . 129
5.8 Implementations of the SAM Algorithm in the Full-Batch Setting . . . 131
5.9 Theoretical Analysis of the Implicit Bias for Diagonal Linear Networks . . . 131
5.9.1 Implicit Bias of the n-SAM Algorithm. . . . 132
5.9.2 Implicit Bias of the 1-SAM Algorithm . . . 134
5.9.3 Comparison between 1-SAM and n-SAM . . . 136
5.10 Convergence of the SAM Algorithm . . . 139
5.10.1 Convergence of Full-Batch n-SAM . . . 139
5.10.2 Convergence of Stochastic SAM . . . 145
5.11 Experimental Details . . . 150
5.12 Additional Deep Learning Experiments . . . 151
5.12.1 The Effect of m in m-SAM . . . 151
5.12.2 The Effect of the Batch Size on SAM . . . 152
5.12.3 The Effect of the Model Width on SAM . . . 152
5.12.4 Sharpness for Models with Batch Normalization . . . 153
5.12.5 Training Loss for ERM vs. SAM Models . . . 153

5.12.6 SAM with a Decreasing Perturbation Radius . . . 154
5.12.7 Experiments with Noisy Labels . . . 154

6 SGD with Large Step Sizes Learns Sparse Features 157
6.1 Preface . . . 157
6.2 Introduction . . . 158
6.2.1 Our Contributions . . . 159
6.2.2 Related Work . . . 159
6.3 The Effective Dynamics of Large Step Size SGD: Sparse Feature Learning 160
6.3.1 Background: SGD is GD with Specific Label Noise . . . 161
6.3.2 The Effective Dynamics Behind Loss Stabilization . . . 162
6.3.3 Sparse Feature Learning . . . 163
6.4 Empirical Evidence of Sparse Feature Learning Driven by SGD . . . 165
6.4.1 Sparse Feature Learning in Diagonal Linear Networks . . . 166
6.4.2 Sparse Feature Learning in Simple ReLU Networks . . . 167
6.4.3 Sparse Feature Learning in Deep ReLU Networks . . . 168
6.5 Conclusions and Insights from our Understanding of the Training Dynamics 170
6.6 SGD and Label Noise GD . . . 172
6.7 Quadratic Parameterization in One Dimension . . . 174
6.8 Empirical Validation of the SDE Modeling . . . 179
6.9 Additional Experimental Results . . . 181

7 A Modern Look at the Relationship between Sharpness and Generalization 185
7.1 Preface . . . 185
7.2 Introduction . . . 186
7.3 Related work . . . 187
7.4 Adaptive Sharpness, its Invariances, and Computation . . . 188
7.4.1 Background on Sharpness . . . 188
7.4.2 Which Invariances Do We Need Sharpness to Capture for Modern Architectures? . . . 190
7.4.3 How to Compute Worst-Case Sharpness Efficiently? . . . 192
7.5 Sharpness vs. Generalization: Modern Setup . . . 192
7.6 Why Doesn't Sharpness Correlate Well with Generalization? . . . 195
7.6.1 The Role of Sharpness in a Controlled Setup . . . 195
7.6.2 Is Sharpness the Right Quantity in the First Place? Insights from Simple Models . . . 198
7.7 Conclusions . . . 200
7.8 Omitted Proofs . . . 206
7.8.1 Asymptotic Analysis of Adaptive Sharpness Measures . . . 206
7.8.2 Derivations for Diagonal Linear Networks . . . 208
7.9 Correlation Between Sharpness and Generalization Gap . . . 210
7.10 ImageNet-1k Models Trained from Scratch from Steiner et al. (2021): Extra Details and Figures . . . 212

7.11 Fine-tuning of ImageNet-1k Models Pretrained on ImageNet-21k from Steiner et al. (2021): Extra Figures and Details 218
7.12 ImageNet Models both Pretrained on ImageNet-1k and ImageNet-21k from Steiner et al. (2021) 223
7.13 Fine-tuning CLIP Models on ImageNet: Extra Details and Figures 226
7.14 Fine-tuning on MNLI: Extra Details and Figures 231
7.15 Training from Scratch on CIFAR-10: Extra Details and Figures 236
7.15.1 The Role of Data Used for Sharpness Evaluation 236
7.15.2 The Role of the Number of Iterations in Auto-PGD 238
7.15.3 The Role of m in m-Sharpness 239
7.15.4 The Role of Different Sharpness Definitions and Radii 241

Conclusions **255**

Bibliography **259**

1 Introduction

Modern deep learning has achieved remarkable progress in many areas: from healthcare (Esteva et al., 2019) and science (Degrave et al., 2022) to conversational models that can solve complex tasks and exhibit non-trivial reasoning capabilities (Bubeck et al., 2023). Despite this progress, it has been observed that *adversarial robustness does not automatically emerge with more data or larger models*. Even state-of-the-art large language models trained on trillions of words can be relatively easily manipulated by using carefully chosen adversarial strings (Zou et al., 2023). This clearly suggests that specialized algorithms like adversarial training are needed to explicitly take into account worst-case examples during training (Madry et al., 2018). Moreover, the existence of adversarial examples can also be seen as a generalization issue and to achieve robustness, it is also necessary to ensure generalization beyond the training set. Thus, we explore two interconnected aspects of modern deep learning in this thesis: *robustness* and *generalization*. To develop a better understanding of generalization, in this thesis, we analyze the training dynamics of gradient-based methods and features learned by deep networks. In addition, we explore the connection between robustness in the input space and parameter space and different notions of overfitting: catastrophic overfitting in adversarial training and overfitting in the overparameterized regime. We start from providing a general overview of the related areas relevant to the works presented in this thesis.

Adversarial robustness. The robustness of machine learning models based on empirical risk minimization to small worst-case perturbations has been a topic of active research for decades (Lowd and Meek, 2005; Globerson and Roweis, 2006). However, the recent success of deep learning has brought renewed interest in this issue, particularly due to the discovery of adversarial examples – tiny input perturbations that can easily deceive state-of-the-art deep neural networks into making incorrect predictions (Szegedy et al., 2014). Adversarial perturbations are formally defined as input modifications that can alter a classifier's prediction while staying within a predefined perturbation set, usually chosen such that the true label of the input should remain unchanged. Robust accuracy, a common robustness measure, represents the proportion of datapoints for which the classifier makes correct predictions for all possible perturbations from this set. Calculating the exact robust accuracy is computationally intractable for most models, so upper bounds are estimated

using adversarial attacks, which attempt to find successful adversarial perturbations. The tightness of these bounds depends on the effectiveness of the attack method used, with suboptimal methods potentially leading to an overestimation of the model's robustness. The threat model, which defines the set of allowed perturbations, is a crucial aspect of robustness evaluation. Throughout the thesis, we will focus on ℓ_p-bounded perturbations (particularly with $p = \infty$) due to their simplicity, although we note that it is clearly not a sufficient definition of robustness (Gilmer et al., 2018).

Implicit regularization. The implicit regularization effect of the piece-wise constant step size schedule introduced in He et al. (2016a), which often leads to a distinct loss stabilization pattern, remains poorly understood. The importance of large step sizes for generalization has been investigated from various angles, such as their potential role in minimizing complexity measures related to the flatness of minima and guiding stochastic gradient descent (SGD) iterates towards flatter minima (Keskar et al., 2016). However, the appropriate flatness measure is often debated, and its relevance to understanding generalization is questionable, as full-batch gradient descent with large step sizes can lead to flat solutions that do not generalize well (Cohen et al., 2021). Stability analysis provides insights into the properties of the minimum that SGD or gradient descent (GD) can potentially reach depending on the step size, but it does not capture the full training dynamics, such as the large step size phase where SGD only converges after the step size is decayed. The implicit bias of SGD augmented with label noise, which shares similarities with the standard noise in SGD, has been characterized, but only in the final stage of training, close to a zero-loss solution set (Li et al., 2022). While the dynamics of GD with large step sizes have received attention, particularly the edge-of-stability phenomenon and the catapult mechanism (Lewkowycz et al., 2020), the absence of stochastic noise in the analysis makes it unsuitable for capturing the behavior of stochastic training. The presence of sparse features and low-rank structures in deep networks trained with large step size SGD has also been observed and utilized in model compression, knowledge distillation, and the lottery ticket hypothesis (Denton et al., 2014; Hinton et al., 2015; Frankle and Carbin, 2018), suggesting that the step size schedule may play a key role in the emergence of such hidden structures.

Robustness in the parameter space. The connection between robustness in the weight space and generalization has been extensively studied. Random weight perturbations are employed in techniques like dropout (Srivastava et al., 2014), and the performance degradation observed when using larger batch sizes for training has been linked to the sharpness of the obtained parameters. This has inspired the concept of minimizing sharpness during training to enhance generalization, leading to methods such as Sharpness-Aware Minimization (SAM) (Foret et al., 2021), which modifies SGD to take gradient steps at worst-case points in the vicinity of the current iterate. Theoretical investigations have also focused on the sharpness properties of minima, particularly in deep linear networks (Mulayoff and Michaeli, 2020), and generalization bounds based on average-case sharpness and quantities related to the optimization trajectory of SGD (Neu, 2021). Studies on the relationship between sharpness and generalization have demonstrated a strong correlation

across a wide range of models and hyperparameter settings (Keskar et al., 2016; Jiang et al., 2019). However, the experimental methodology has been criticized for potentially masking failures of generalization measures and should be evaluated within the framework of distributional robustness (Dziugaite et al., 2020). The notion that flat minima can benefit generalization has motivated various methods that optimize for more robust minima, using criteria ranging from random perturbations, such as in Entropy-SGD (Chaudhari et al., 2016), to worst-case perturbations, as in SAM and its variations (Kwon et al., 2021). Simultaneously, research on the implicit bias of SGD suggests an implicit minimization of hidden complexity measures related to the flatness of minima, with works focusing on the implicit regularization of SGD in terms of the gradient norm and the trace of the Hessian matrix (Xing et al., 2018). Recent works have focused on sharpness-related quantities based on the Hessian matrix, such as the maximum eigenvalue and its relation to the learning rate in full-batch gradient descent, with the shared goal of using sharpness-related metrics to gain a deeper understanding of optimization and generalization in deep networks (Arora et al., 2022; Damian et al., 2023).

Research questions of the thesis. It is very important to establish strong foundations and rigorous understanding to be able to reliably build on top of the current deep learning algorithms. We believe that many aspects of deep learning can be understood empirically via carefully designed experiments and theoretically via simple models. With this in mind, we focus on the following research questions throughout the thesis:

- *Why do computationally efficient adversarial training methods fail and how to resolve this?* Standard adversarial training adds a significant computational overhead since it requires computing adversarial examples on-the-fly using many iterations of projected gradient descent. Instead, using only a few gradient steps is key to unlock practical adversarial training at scale. However, this leads to a failure mode known as *catastrophic overfitting* where the model completely loses its robustness after some point in training.
- *How to evaluate robustness in a more accurate and standardized way, without resorting to adaptive adversarial attacks?* It is well accepted that *adaptive attacks*, i.e., attacks tailored specifically to a particular defense, are gold standard for robustness evaluation (Tramèr et al., 2020). However, it is infeasible to evaluate adaptively all new defenses against adversarial examples since adaptive attacks require a lot of manual trial and error. Thus, we also need some reliable automated attacks, ideally that are not based on gradients. This is important since many defenses are based on *gradient obfuscation* that only prevents gradient-based attacks but does not inherently improve adversarial robustness.
- *What does robustness in the parameter space imply for generalization? How can we understand it via analysis of the training dynamics?* Robustness in the parameter space or sharpness has been linked to generalization, with the intuition that a model that is robust in the parameter space is likely to generalize well to unseen data. However, it is unclear how accurate this intuition is since it has never been verified

beyond small-scale settings. Moreover, by analyzing the training dynamics, i.e., the trajectory of the model's parameters during optimization, one can gain insights into what methods like SAM implicitly do. Understanding these aspects can help us design better optimization algorithms and regularization techniques that improve generalization.

- *Why certain optimization algorithms result in better generalization and the learning of qualitatively distinct features?* We are primarily interested in two specific optimization algorithms: SGD with large step sizes and SAM. SGD with large step sizes has been shown to implicitly regularize the model, but it is not clear what this implicit regularization implies on the features learned by the deep network. By studying these optimization algorithms and drawing inspiration from a comprehensive understanding of the training dynamics on simple models, one can gain a deeper understanding of how the choice of optimization algorithm can influence the structure of the learned representations.

1.1 Outline of the thesis

The thesis is divided into two parts: *Robustness in Modern Deep Learning* and *Generalization in Modern Deep Learning*. The first part is based on the following papers:

- Chapter 2 is based on Andriushchenko and Flammarion (2020) (NeurIPS 2020) where we focus on the problem of computationally efficient adversarial training for deep learning models. In particular, we analyze the phenomenon of catastrophic overfitting, which occurs when the model quickly loses its robustness over a single epoch of training.
- Chapter 3 is based on Andriushchenko et al. (2020) (ECCV 2020) where we focus on the problem of adversarial attacks on image classification models in the black-box setting, where the attacker has no access to the model's parameters or gradients.
- Chapter 4 is based on Croce et al. (2021) (NeurIPS 2021 Datasets and Benchmarks Track) where we focus on the problem of benchmarking adversarial robustness of machine learning models. We introduce a standardized benchmark for adversarial robustness, `RobustBench`, which aims to provide a reliable evaluation of the robustness of the considered models within a reasonable computational budget.

The second part of the thesis is based on the following papers:

- Chapter 5 is based on Andriushchenko and Flammarion (2022) (ICML 2022) where we focus on the recent training method called Sharpness-Aware Minimization (SAM) which has been shown to significantly improve generalization in various settings. We argue that the existing justifications for the success of SAM which are based on a PAC-Bayes generalization bound and the idea of convergence to flat minima are

incomplete. Via a set of experiments and theoretical results, we provide a new perspective on the reasons behind better generalization of SAM.

- Chapter 6 is based on Andriushchenko et al. (2023d) (ICML 2023) where we present a study of the dynamics of the Stochastic Gradient Descent (SGD) in the training of neural networks. We show that commonly used large step sizes may lead the iterates to jump from one side of a valley to the other causing *loss stabilization*, and this stabilization induces a hidden stochastic dynamics that *biases it implicitly* toward sparse predictors.

- Chapter 7 is based on Andriushchenko et al. (2023b) (ICML 2023) where we provide a comprehensive study of how well different definitions of sharpness of minima correlate with generalization. We conclude that sharpness does not necessarily correlate well with generalization but rather with some training parameters like the learning rate.

1.2 Contributions beyond this thesis

In addition to the works presented in the thesis, the author contributed to the following papers.

Contributions in adversarial robustness. In Croce et al. (2020) (AAAI 2022), we focus on improving *black-box* attacks to make robustness evaluations for ℓ_p-bounded and sparse perturbations more reliable. Moreover, in Kireev et al. (2021) (UAI 2022), we study the effect of different adversarial training schemes on the performance on natural image corruptions. We find that even standard ℓ_p adversarial training can be an effective technique against such corruptions and proposed an improved training scheme motivated by adversarial training with perceptual distances. In Andriushchenko et al. (2022) (CVPR 2022 workshop), we study the task of adversarially robust image attribution for content provenance as part of the Content Authenticity Initiative that aims to fight disinformation and fake content on the internet. In Kireev et al. (2023) (NeurIPS 2023), we explore more plausible *cost-aware* threat models for tabular datasets. In Debenedetti et al. (2023) (ICLR 2024 workshop), we study scaling laws for adversarial robustness concluding that simply scale the model size has very limited gains. Finally, in Andriushchenko (2023), we explore the feasibility of adversarial attacks on GPT-4 using a random search technique similar to the one discussed in this thesis (Andriushchenko et al., 2020).

Contributions in understanding generalization and training dynamics. In Mosbach et al. (2021) (ICLR 2021), we provide reasons behind fine-tuning instability of BERT language models. More recently, in Andriushchenko et al. (2023a) (NeurIPS 2023), we develop a better understanding of the effect of sharpness-aware minimization on the features learned by the model. In Shin et al. (2023), we study the effect of overparameterization on convergence and generalization of sharpness-aware minimization. In the recent work Andriushchenko et al. (2023c) (NeurIPS 2023 workshop), we study the role of weight decay on generalization of overparameterized deep networks and on optimization and training

stability of LLMs. In Adilova et al. (2024) (ICLR 2024), we study the phenomenon of layerwise linear mode connectivity and discuss interesting connections to federated learning. The most recent work is Zhao et al. (2024) (ICLR 2024 workshop) where we study instruction fine-tuning of LLMs and suggest a very simple but competitive baseline for selection of the data points.

Part I
Robustness in Modern Deep Learning

2 Understanding and Improving Fast Adversarial Training

2.1 Preface

In this chapter, based on Andriushchenko and Flammarion (2020), we focus on the problem of computationally efficient adversarial training for deep learning models. In particular, we analyze the phenomenon of catastrophic overfitting, which occurs when the model quickly loses its robustness over a single epoch of training.

Summary A recent line of work focused on making adversarial training computationally efficient for deep learning models. In particular, Wong et al. (2020) showed that ℓ_∞-adversarial training with fast gradient sign method (FGSM) can fail due to a phenomenon called *catastrophic overfitting*, when the model quickly loses its robustness over a single epoch of training. We show that adding a random step to FGSM, as proposed in Wong et al. (2020), does not prevent catastrophic overfitting, and that randomness is not important per se — its main role being simply to reduce the magnitude of the perturbation. Moreover, we show that catastrophic overfitting is not inherent to deep and overparametrized networks, but can occur in a single-layer convolutional network with a few filters. In an extreme case, even *a single filter* can make the network highly non-linear *locally*, which is the main reason why FGSM training fails. Based on this observation, we propose a new regularization method, `GradAlign`, that *prevents catastrophic overfitting* by explicitly maximizing the gradient alignment inside the perturbation set and improves the quality of the FGSM solution. As a result, `GradAlign` allows to successfully apply FGSM training also for larger ℓ_∞-perturbations and reduce the gap to multi-step adversarial training. The code of our experiments is available at https://github.com/tml-epfl/understanding-fast-adv-training.

Co-authors Nicolas Flammarion.

Contributions Maksym Andriushchenko made key contributions to all aspects of the project.

2.2 Introduction

Machine learning models based on empirical risk minimization are known to be often non-robust to small worst-case perturbations. For decades, this has been the topic of active research by the statistics, optimization and machine learning communities (Huber, 1981; Ben-Tal et al., 2009; Globerson and Roweis, 2006; Biggio and Roli, 2018b). However, the recent success of deep learning (LeCun et al., 2015; Schmidhuber, 2015) has raised the interest in this topic. The lack of robustness in deep learning is clearly illustrated by the existence of *adversarial examples*, i.e. tiny input perturbations that can easily fool state-of-the-art deep neural networks into making wrong predictions (Szegedy et al., 2014; Goodfellow et al., 2015).

Improving the robustness of machine learning models is motivated not only from the security perspective (Biggio and Roli, 2018b). Adversarially robust models have better interpretability properties (Tsipras et al., 2019; Santurkar et al., 2019) and can generalize better (Zhu et al., 2019; Bochkovskiy et al., 2020) including also improved performance under some distribution shifts (Xie et al., 2020) (although on some performing worse, see Taori et al. (2020b)). In order to improve the robustness, two families of solutions have been developed: *adversarial training* (AT) that amounts to training the model on adversarial examples (Goodfellow et al., 2015; Madry et al., 2018) and *provable defenses* that derive and optimize robustness certificates (Wong and Kolter, 2018; Raghunathan et al., 2018; Cohen et al., 2019). Currently, adversarial-training based methods appear to be preferred by practitioners since they (a) achieve higher empirical robustness (although without providing a robustness certificate), (b) can be scaled to state-of-the-art deep networks without affecting the inference time (unlike smoothing-based approaches (Cohen et al., 2019)), and (c) work equally well for different threat models. Adversarial training can be formulated as a robust optimization problem (Shaham et al., 2018; Madry et al., 2018) which takes the form of a non-convex non-concave min-max problem. However, computing the optimal adversarial examples is an NP-hard problem (Katz et al., 2017; Weng et al., 2018). Thus adversarial training can only rely on approximate methods to solve the inner maximization problem.

One popular approximation method successfully used in adversarial training is the PGD attack (Madry et al., 2018) where multiple steps of projected gradient descent are performed. It is now widely believed that models adversarially trained via the PGD attack (Madry et al., 2018; Zhang et al., 2019c) are robust since small adversarially trained networks can be formally verified (Carlini et al., 2017; Tjeng et al., 2019b; Wong et al., 2020), and larger models could not be broken on public challenges (Madry et al., 2018; Zhang et al., 2019c). Recently, (Croce and Hein, 2020b) evaluated the majority of recently published defenses to conclude that the standard ℓ_∞ PGD training achieves the best empirical robustness; a result which can only be improved using semi-supervised approaches (Hendrycks et al., 2019; Alayrac et al., 2019; Carmon et al., 2019). In contrast, other empirical defenses that were claiming improvements over standard PGD training had overestimated the robustness of their reported models (Croce and Hein, 2020b). These experiments imply that

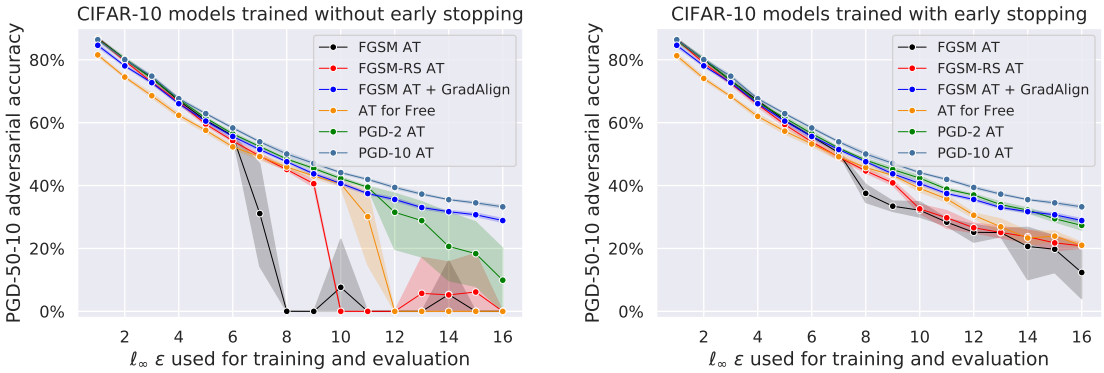


Figure 2.1: Robustness of different adversarial training (AT) methods on CIFAR-10 with ResNet-18 trained and evaluated with different l_∞-radii. The results are averaged over 5 random seeds used for training and reported with the standard deviation. **FGSM AT**: standard FGSM AT, **FGSM-RS AT**: FGSM AT with a random step (Wong et al., 2020), **FGSM AT + GradAlign**: FGSM AT combined with our proposed regularizer `GradAlign`, **AT for Free**: recently proposed method for fast PGD AT (Shafahi et al., 2019), **PGD-2/PGD-10 AT**: AT with a 2-/10-step PGD-attack. Our proposed regularizer `GradAlign` prevents *catastrophic overfitting* in FGSM training and leads to significantly better results which are close to the computationally demanding PGD-10 AT.

adversarial training in general is the key algorithm for robust deep learning, and thus that performing it efficiently is of paramount importance.

Another approximation method for adversarial training is the *Fast Gradient Sign Method* (FGSM) (Goodfellow et al., 2015) which is based on the linear approximation of the neural network loss function. However, the literature is still ambiguous about the performance of FGSM training, i.e. it remains unclear whether FGSM training can *consistently* lead to robust models. For example, (Madry et al., 2018) and (Tramèr et al., 2018) claim that FGSM training works only for small ℓ_∞-perturbations, while (Wong et al., 2020) suggest that FGSM training can lead to robust models for arbitrary ℓ_∞-perturbations if one adds uniformly random initialization before the FGSM step. Related to this, (Wong et al., 2020) further identified a phenomenon called *catastrophic overfitting* where FGSM training first leads to *some* robustness at the beginning of training, but then suddenly becomes non-robust within a single training epoch. However, the reasons for such a failure remain unknown. This motivates us to consider the following question as the main theme of the paper:

When and why does fast adversarial training with FGSM lead to robust models?

Contributions. We first show that not only FGSM training is prone to *catastrophic overfitting*, but the recently proposed fast adversarial training methods (Shafahi et al., 2019; Wong et al., 2020) as well (see Fig. 2.1). We then analyze the reasons why using a random step in FGSM (Wong et al., 2020) helps to slightly mitigate catastrophic overfitting and show it simply boils down to reducing the average magnitude of the perturbations. Then we discuss the connection behind catastrophic overfitting and local linearity in deep networks and in single-layer convolutional networks where we show that even *a*

single filter can make the network non-linear *locally*, and causes the failure of FGSM training. We additionally provide for this case a theoretical explanation which helps to explain why FGSM AT is successful at the beginning of the training. Finally, we propose a regularization method, `GradAlign`, that *prevents catastrophic overfitting* by explicitly maximizing the gradient alignment inside the perturbation set and therefore improves the quality of the FGSM solution. We compare `GradAlign` to other adversarial training schemes in Fig. 2.1 and point out that among all fast adversarial training methods considered only FGSM + `GradAlign` does not suffer from catastrophic overfitting and leads to high robustness even for large ℓ_∞-perturbations.

2.3 Problem overview and related work

Let $\ell(x, y; \theta)$ denote the loss of a ReLU-network parametrized by $\theta \in \mathbb{R}^m$ on the example $(x, y) \sim D$ where D is the data generating distribution.[1] Previous works (Shaham et al., 2018; Madry et al., 2018) formalized the goal of training adversarially robust models as the following robust optimization problem:

$$\min_{\theta} \mathbb{E}_{(x,y)\sim D} \left[\max_{\delta \in \Delta} \ell(x + \delta, y; \theta) \right]. \tag{2.1}$$

We focus here on the ℓ_∞ threat model, i.e. $\Delta = \{\delta \in \mathbb{R}^d, \|\delta\|_\infty \leq \varepsilon\}$, where the adversary can change each input coordinate x_i by at most ε. Unlike classical stochastic saddle point problems of the form $\min_\theta \max_\delta \mathbb{E}[\ell(\theta, \delta)]$ (Juditsky et al., 2011), the inner maximization problem here is inside the expectation. Therefore the solution of each subproblem $\max_{\delta\in\Delta} \ell(x + \delta, y; \theta)$ depends on the particular example (x, y) and standard algorithms such as gradient descent-ascent which alternate gradient descent in θ and gradient ascent in δ cannot be used. Instead each of these *non-concave* maximization problems has to be solved independently. Thus, an inherent trade-off appears between computationally efficient approaches which aim at solving this inner problem in as few iterations as possible and approaches which aim at solving the problem more accurately but with more iterations. In an extreme case, the PGD attack (Madry et al., 2018) uses multiple steps of projected gradient ascent (PGD), which is accurate but computationally expensive. At the other end of the spectrum, Fast Gradient Sign Method (FGSM) (Goodfellow et al., 2015) performs *only* one iteration of gradient ascent with respect to the ℓ_∞-norm:

$$\delta_{FGSM} \stackrel{\text{def}}{=} \varepsilon \operatorname{sign}(\nabla_x \ell(x, y; \theta)), \tag{2.2}$$

followed by a projection of $x + \delta_{FGSM}$ onto the $[0, 1]^d$ to ensure it is a valid input.[2] This leads to a fast algorithm which, however, does not always lead to robust models as observed in (Madry et al., 2018; Tramèr et al., 2018). A closer look at the evolution of the robustness during FGSM AT reveals that using FGSM can lead to a model with some degree of robustness but only until a point where the robustness suddenly drops.

[1] In practice we use training samples with random data augmentation.

[2] Throughout the paper we will focus on image classification, i.e. inputs x will be images.

This phenomenon is called *catastrophic overfitting* in Wong et al. (2020). As a partial solution, the training can be stopped just before that point which leads to non-trivial but suboptimal robustness as illustrated in Fig. 2.1. Wong et al. (2020) further notice that initializing FGSM from a random starting point $\eta \sim \mathcal{U}([-\varepsilon, \varepsilon]^d)$, i.e. using the following perturbation where $\Pi_{[-\varepsilon,\varepsilon]^d}$ denotes the projection:

$$\delta_{FGSM-RS} \stackrel{\text{def}}{=} \Pi_{[-\varepsilon,\varepsilon]^d}[\eta + \alpha \operatorname{sign}(\nabla_x \ell(x + \eta, y; \theta))], \tag{2.3}$$

helps to mitigate catastrophic overfitting and leads to better robustness for the considered ε values (e.g. $\varepsilon = 8/255$ on CIFAR-10 in Wong et al. (2020)). Along the same lines, Vivek and Babu (2020) observe that using dropout on all layers (including convolutional) also helps to stabilize FGSM AT.

An alternative solution is to interpolate between FGSM and PGD AT. For example, (Wang et al., 2019b) suggest to first use FGSM AT, and later to switch to multi-step PGD AT which is motivated by their analysis suggesting that the inner maximization problem has to be solved more accurately at the end of training. Shafahi et al. (2019) propose to run PGD with step size $\alpha = \varepsilon$ and simultaneously update the weights of the network. On a related note, Zhang et al. (2019a) collect the weight updates during PGD, but apply them after PGD is completed. Additionally, Zhang et al. (2019a) update the gradients of the first layer multiple times. However, none of these approaches are conclusive, either leading to comparable robustness to FGSM-RS training (Wong et al., 2020) and still failing for higher ℓ_∞-radii (see Fig. 2.1 for Shafahi et al. (2019) and Wong et al. (2020)) or being in the worst case as expensive as multi-step PGD AT (Wang et al., 2019b). Additionally, some previous works deviate from the robust optimization formulation stated in Eq. (2.1) and instead regularize the model to improve robustness (Simon-Gabriel et al., 2019; Moosavi-Dezfooli et al., 2019b; Qin et al., 2019), however this does not lead to higher robustness compared to standard adversarial training. We focus next on analyzing the FGSM-RS training (Wong et al., 2020) as the other recent variations of fast adversarial training (Shafahi et al., 2019; Zhang et al., 2019a; Vivek and Babu, 2020) lead to models with similar robustness.

Experimental setup. Unless mentioned otherwise, we perform training on PreAct ResNet-18 (He et al., 2016b) with the cyclic learning rates (Smith, 2017) and half-precision training (Micikevicius et al., 2018) following the setup of Wong et al. (2020). We evaluate adversarial robustness using the PGD-50-10 attack, i.e., with 50 iterations and 10 restarts with step size $\alpha = \varepsilon/4$ following Wong et al. (2020). More experimental details are specified in Appendix 2.9.

2.4 The role and limitations of using random initialization in FGSM training

First, we show that FGSM with a random step fails to resolve catastrophic overfitting for larger ε. Then we provide evidence against the explanation given by Wong et al. (2020) on the benefit of randomness for FGSM AT, and propose a new explanation based on the linear approximation quality of FGSM.

FGSM with random step does not resolve catastrophic overfitting. Crucially, Wong et al. (2020) observed that adding an initial random step to FGSM as in Eq. (2.3) helps to avoid catastrophic overfitting. However, this holds only if the step size is not too large (as illustrated in Fig. 3 of Wong et al. (2020) for $\varepsilon = 8/255$) and, more importantly, only for small enough ε as we show in Fig. 2.1. Indeed, using the step size $\alpha = 1.25\varepsilon$ recommended by Wong et al. (2020) *extends* the working regime of FGSM but only from $\varepsilon = 6/255$ to $\varepsilon = 9/255$, with 0% adversarial accuracy for $\varepsilon = 10/255$. When early stopping is applied (Fig. 2.1, right), there is still a significant gap compared to PGD-10 training, particularly for large ℓ_∞-radii. For example, for $\varepsilon = 16/255$, FGSM-RS AT leads to 22.24% PGD-50-10 accuracy while PGD-10 AT obtains a much better accuracy of 30.65%.

Previous explanation: randomness diversifies the threat model. A hypothesis stated in Wong et al. (2020) was that FGSM-RS helps to avoid catastrophic overfitting by diversifying the threat model. Indeed, the random step allows to have perturbations not only at the corners $\{-\varepsilon, \varepsilon\}^d$ like the FGSM-attack[3], but rather in the whole ℓ_∞-ball, $[-\varepsilon, \varepsilon]^d$. Here we refute this hypothesis by modifying the usual PGD training by projecting onto $\{-\varepsilon, \varepsilon\}^d$ the perturbation obtained via the PGD attack. We perform experiments on CIFAR-10 with ResNet-18 with ℓ_∞-perturbations of radius $\varepsilon = 8/255$ over 5 random seeds. FGSM AT leads to catastrophic overfitting achieving $0.00 \pm 0.00\%$ adversarial accuracy if early stopping is not applied, while the standard PGD-10 AT and our modified PGD-10 AT schemes achieve $50.48 \pm 0.20\%$ and $50.64 \pm 0.23\%$ adversarial accuracy respectively. Thereby similar robustness as the original PGD AT can still be achieved without training on pertubations from the interior of the ℓ_∞-ball. We conclude that *diversity* of adversarial examples is not crucial here. What makes the difference is rather having an iterative instead of a single-step procedure to find a corner of the ℓ_∞-ball that sufficiently maximizes the loss.

New explanation: a random step improves the linear approximation quality. Using a random step in FGSM is *guaranteed* to decrease the expected magnitude of the perturbation. This simple observation is formalized in the following lemma.

Lemma 1. (Effect of the random step) *Let $\eta \sim \mathcal{U}([-\varepsilon, \varepsilon]^d)$ be a random starting point, and $\alpha \in [0, 2\varepsilon]$ be the step size of FGSM-RS defined in Eq. (2.3), then*

$$\mathbb{E}_\eta \left[\|\delta_{FGSM-RS}(\eta)\|_2 \right] \leq \sqrt{\mathbb{E}_\eta \left[\|\delta_{FGSM-RS}(\eta)\|_2^2 \right]} = \sqrt{d} \sqrt{-\frac{1}{6\varepsilon}\alpha^3 + \frac{1}{2}\alpha^2 + \frac{1}{3}\varepsilon^2}. \quad (2.4)$$

[3]For simplicity, we ignore the projection of $x + \delta$ onto $[0, 1]^d$ in this section.

The proof is deferred to Appendix 2.8.1. We first remark that the upper bound is in the range $[1/\sqrt{3}\sqrt{d}\varepsilon, \sqrt{d}\varepsilon]$, and therefore always less or equal than $\|\delta_{FGSM}\|_2 = \sqrt{d}\varepsilon$. We visualize our bound in Fig. 2.2 where the expectation is approximated by Monte-Carlo sampling over 1,000 samples of η, and note that the bound becomes increasingly tight for high-dimensional inputs.

The key observation here is that among all possible perturbations of ℓ_∞-norm ε, perturbations with a smaller ℓ_2-norm benefit from a better linear approximation. This statement follows from the second-order Taylor expansion for twice differentiable functions:

$$f(x+\delta) \approx f(x) + \langle \nabla_x f(x), \delta \rangle + \left\langle \delta, \nabla^2_{xx} f(x) \delta \right\rangle,$$

i.e. a smaller value of $\|\delta\|_2^2$ implies a smaller linear approximation error $|f(x+\delta) - f(x) - \langle \nabla_x f(x), \delta \rangle|$. Moreover, the same property still holds empirically for the non-differentiable ReLU networks (see Appendix 2.10.1). We conclude that by reducing in expectation the length of the perturbation $\|\delta\|_2$, the FGSM-RS approach of Wong et al. (2020) takes advantage of a better linear approximation. This is supported by the fact that FGSM-RS AT also leads to catastrophic overfitting if the step size α is chosen to be too large (see Fig. 3 in Wong et al. (2020)), thus providing no benefits over FGSM AT even when combined with early stopping. We argue this is the main improvement over the standard FGSM AT.

Successful FGSM AT does not require randomness. If having perturbation with a too large ℓ_2-norm is indeed the key factor in catastrophic overfitting, we can expect that just reducing the step size of the standard FGSM should work equally well as FGSM-RS. For $\varepsilon = 8/255$ on CIFAR-10, Wong et al. (2020) recommend to use FGSM-RS with step size $\alpha = 1.25\varepsilon$ which induces a perturbation of expected ℓ_2-norm $\|\delta_{FGSM-RS}\|_2 \approx 7/255\sqrt{d}$. This corresponds to using standard FGSM with a step size $\alpha \approx 7/255$ instead of $\alpha = \varepsilon = 8/255$ (see the dashed line in Fig. 2.2). We report the results in Table 2.1 and observe that simply reducing the step size of FGSM (without *any* randomness) leads to the same level of robustness. We show further in Fig. 2.3 that when used with a smaller step size, the

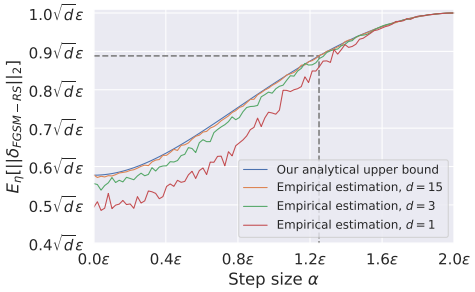


Figure 2.2: Visualization of our upper bound on $\mathbb{E}_\eta[\|\delta_{FGSM-RS}\|_2]$. The dashed line corresponds to the step size $\alpha = 1.25\varepsilon$ recommended in Wong et al. (2020).

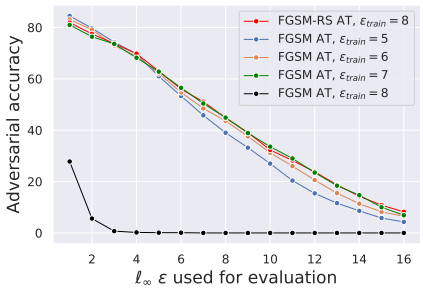


Figure 2.3: Robustness of FGSM-trained ResNet-18 on CIFAR-10 with different ε_{train} used for training compared to FGSM-RS AT with $\varepsilon_{train} = 8/255$.

Table 2.1: Robustness of FGSM AT with a reduced step size ($\alpha = 7/255$) compared to the FGSM-RS AT proposed in Wong et al. (2020) ($\alpha = 10/255$) for $\varepsilon = 8/255$ on CIFAR-10 for ResNet-18 trained with early stopping. The results are averaged over 5 random seeds used for training.

Accuracy \ Model	FGSM AT	FGSM $\alpha = 7/255$ AT	FGSM-RS AT
PGD-50-10	$36.35 \pm 1.74\%$	$45.35 \pm 0.48\%$	$45.60 \pm 0.19\%$

robustness of *standard* FGSM training even without early stopping can generalize to much higher ε. This contrasts with the previous literature (Madry et al., 2018; Tramèr et al., 2018). We conclude from these experiments that a more direct way to improve FGSM AT and to prevent it from catastrophic overfitting is to simply reduce the step size. Note that this still leads to suboptimal robustness compared to PGD AT (see Fig. 2.1) for ε larger than the one used during training, since in this case adversarial examples can only be generated inside the smaller ℓ_∞-ball. This motivates us to take a closer look on *how* and *why* catastrophic overfitting occurs to be able to prevent it without reducing the FGSM step size.

2.5 Understanding catastrophic overfitting via gradient alignment

First, we establish a connection between catastrophic overfitting and local linearity of the model. Then we show that catastrophic overfitting also occurs in a single-layer convolutional network, for which we analyze local linearity both empirically and theoretically.

When can the inner maximization problem be accurately solved with FGSM? Recall that the FGSM attack (Goodfellow et al., 2015) is obtained as a closed-form solution of the following optimization problem: $\delta_{FGSM} = \arg\max_{\|\delta\|_\infty \leq \varepsilon} \langle \nabla_x \ell(x, y; \theta)), \delta \rangle$. Thus, the FGSM attack is guaranteed to find the optimal adversarial perturbation if $\nabla_x \ell(x, y; \theta)$ is constant inside the ℓ_∞-ball around the input x, i.e. the loss function is *locally linear*. This motivates us to study the evolution of local linearity during FGSM training and its connection to catastrophic overfitting. With this aim, we define the following local linearity metric of the loss function ℓ:

$$\mathbb{E}_{(x,y)\sim D,\ \eta\sim\mathcal{U}([-\varepsilon,\varepsilon]^d)} \left[\cos\left(\nabla_x \ell(x, y; \theta), \nabla_x \ell(x + \eta, y; \theta)\right)\right], \tag{2.5}$$

which we refer to as *gradient alignment*. This quantity is easily interpretable: it is equal to one for models linear inside the ℓ_∞-ball of radius ε, and it is approximately zero when the input gradients are nearly orthogonal to each other. Previous works also considered local linearity of deep networks (Moosavi-Dezfooli et al., 2019b; Qin et al., 2019), however rather with the goal of introducing regularization methods that improve robustness as an *alternative* to adversarial training. More precisely, Moosavi-Dezfooli et al. (2019b) propose to use a curvature regularization method that uses the FGSM point, and Qin et al. (2019) find the input point where local linearity is maximally violated using an

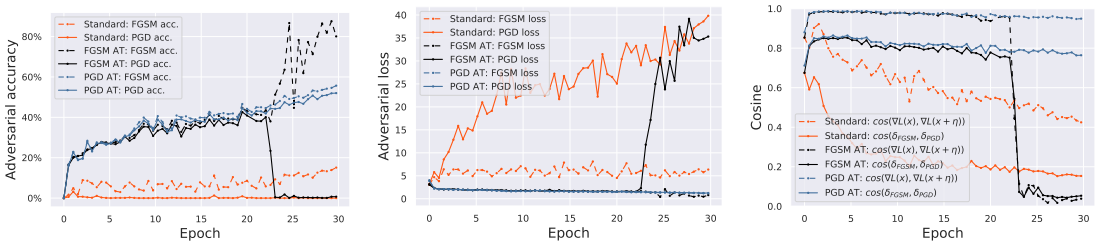


Figure 2.4: Visualization of the training process of standardly trained, FGSM trained, and PGD-10 trained ResNet-18 on CIFAR-10 with $\varepsilon = 8/255$. All the statistics are calculated on the test set. Catastrophic overfitting for the FGSM AT model occurs around epoch 23 and is characterized by a sudden drop in the PGD accuracy, a gap between the FGSM and PGD losses, and a dramatic decrease of *local linearity*.

iterative method, leading to comparable computational cost as PGD AT. In contrast, we analyze here gradient alignment to improve FGSM training without seeking an alternative to it.

Catastrophic overfitting in deep networks. To understand the link between catastrophic overfitting and local linearity, we plot in Fig. 2.4 the adversarial accuracies and the loss values obtained by FGSM and PGD AT on CIFAR-10 using ResNet-18, together with the gradient alignment (see Eq. 2.5) and the cosine between FGSM and PGD perturbations. We compute these statistics on the test set. Catastrophic overfitting occurs for FGSM AT around epoch 23, and is characterized by the following intertwined events: (a) There is a *sudden drop* in the PGD accuracy from 40.1% to 0.0%, along with an *abrupt jump* of the FGSM accuracy from 43.5% to 86.7%. In contrast, before the catastrophic overfitting, the ratio between the average PGD and FGSM losses never exceeded 1.05. This suggests that FGSM cannot anymore accurately solve the inner maximization problem. (b) Concurrently, after catastrophic overfitting, the gradient alignment of the FGSM model experiences a *phase transition* and drops significantly from 0.95 to 0.05 within an epoch of training, i.e. *the input gradients become nearly orthogonal inside the ℓ_∞-ball*. We observe the same drop also for $\cos(\delta_{FGSM}, \delta_{PGD})$ which means that the FGSM and PGD directions are not aligned anymore (as also observed in Tramèr et al. (2018)). This echoes the observation made in Nakkiran et al. (2019) that SGD on the standard loss of a neural network learns models of increasing complexity. We observe qualitatively the same phenomenon for FGSM AT, where the complexity is captured by the degree of local non-linearity. The connection between local linearity and catastrophic overfitting sparks interest for a further analysis in a simpler setting.

Catastrophic overfitting in a single-layer CNN. We show that catastrophic overfitting is not inherent to deep and overparametrized networks, and can be observed in a very simple setup. For this we train a single-layer CNN with four filters on CIFAR-10 using FGSM AT with $\varepsilon = 10/255$ (see Sec. 2.9 for details). We observe that catastrophic overfitting occurs in this simple model as well, and its pattern is the same as in ResNet: a simultaneous drop of the PGD accuracy and gradient alignment (see Appendix 2.10.2). The advantage of considering a simple model is that we can inspect the learned filters

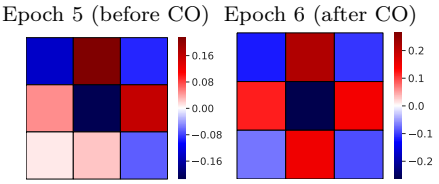


Figure 2.5: Filter w_4 (green channel) in a single-layer CNN before and after catastrophic overfitting (CO).

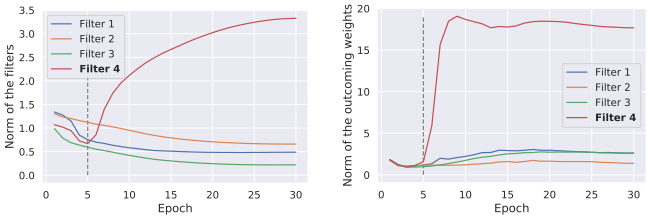


Figure 2.6: Evolution of the weight norms in a single-layer CNN before and after catastrophic overfitting (dashed line).

and understand what causes the network to become highly non-linear locally. We observe that after catastrophic overfitting the network has learned in filter w_4 a variant of the Laplace filter (see Fig. 2.5), an edge-detector filter which is well-known for *amplifying high-frequency noise* such as uniform noise (Gonzales and Woods, 2002). Until the end of training, filter w_4 preserves its direction (see Appendix 2.10.2 for detailed visualizations), but grows significantly in its magnitude together with its outcoming weights, in contrast to the rest of the filters as shown in Fig. 2.5. Interestingly, if we set w_4 to zero, the network largely *recovers local linearity*: the gradient alignment increases from 0.08 to 0.71, recovering its value before catastrophic overfitting. Thus, in this extreme case, *even a single convolutional filter can cause catastrophic overfitting*. Next we analyze formally gradient alignment in a single-layer CNN and elaborate on the connection to the noise sensitivity.

Analysis of gradient alignment in a single-layer CNN. We analyze here a single-layer CNN with ReLU-activation. Let $Z \in \mathbb{R}^{p\times k}$ be the matrix of k non-overlapping image patches extracted from the image $x = (Z) \in \mathbb{R}^d$ such that $z_j = z_j(x) \in \mathbb{R}^p$. The model prediction f is parametrized by $(W, b, U, c) \in \mathbb{R}^{p\times m} \times \mathbb{R}^m \times \mathbb{R}^{m\times k} \times \mathbb{R}$, and its prediction and the input gradient are given as

$$f(x) = \sum_{i=1}^{m}\sum_{j=1}^{k} u_{ij} \max\{\langle w_i, z_j\rangle + b_i, 0\} + c, \quad \nabla_x f(x) = \left(\sum_{i=1}^{m}\sum_{j=1}^{k} u_{ij} \mathbb{1}_{\langle w_i, z_j\rangle + b_i \geq 0} w_i e_j^T\right).$$

We observe that catastrophic overfitting only happens at later stages of training. At the beginnning of the training, the gradient alignment is very high (see Fig. 2.4 and Fig. 2.11), and FGSM solves the inner maximization problem accurately enough. Thus, an important aspect of FGSM training is that the model starts training from *highly aligned gradient*. This motivates us to inspect closely gradient alignment at initialization.

Lemma 2. (Gradient alignment at initialization) *Let $z \sim \mathcal{U}([0,1]^p)$ be an image patch for $p \geq 2$, $\eta \sim \mathcal{U}([-\varepsilon, \varepsilon]^d)$ a point inside the ℓ_∞-ball, the parameters of a single-layer CNN initialized i.i.d. as $w \sim \mathcal{N}(0, \sigma_w^2 I_p)$ for every column of W, $u \sim \mathcal{N}(0, \sigma_u^2 I_m)$ for every column of U, $b := 0$, then the gradient alignment is lower bounded by*

$$\lim_{k,m\to\infty} \cos\left(\nabla_x \ell(x, y), \nabla_x \ell(x + \eta, y)\right) \geq \max\left\{1 - \sqrt{2}\, \mathbb{E}_{w,z}\left[e^{-\frac{1}{\varepsilon^2}\langle w/\|w\|_2, z\rangle^2}\right]^{1/2}, 0.5\right\}.$$

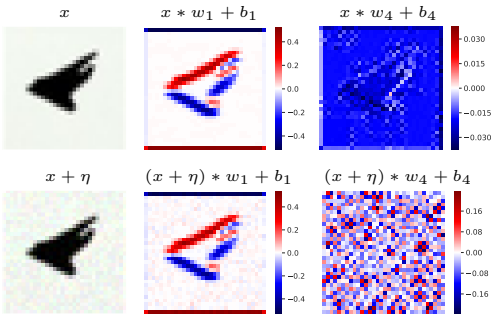


Figure 2.7: Feature maps of filters w_1 and w_4 in a single-layer CNN. A small noise η is significantly amplified by the Laplace filter w_4 in contrast to a regular filter w_1.

The lemma implies that for randomly initialized CNNs with a large enough number of image patches k and filters m, gradient alignment cannot be smaller than 0.5. This is in contrast to the value of 0.12 that we observe after catastrophic overfitting when the weights are no longer i.i.d. We note that the lower bound of 0.5 is quite pessimistic since it holds for an arbitrarily large ε. The lower bound is close to 1 when ε is small compared to $\mathbb{E}\,\|z\|_2$ which is typical in adversarial robustness (see Appendix 2.8.2 for the visualization of the lower bound). High gradient alignment at initialization also holds empirically for deep networks as well, e.g. for ResNet-18 (see Fig. 2.4), starting from the value of 0.85 in contrast to 0.04 after catastrophic overfitting. Thus, it appears to be a general phenomenon that the standard initialization scheme of neural network weights (He et al., 2015) ensures the *initial* success of FGSM training.

In contrast, after some point during training, the network can learn parameters which lead to a significant reduction of gradient alignment. For simplicity, let us consider a single-filter CNN where the gradient alignment for a filter w and bias b at points x and $x + \eta$ has a simple expression:

$$\cos\left(\nabla_x \ell(x, y), \nabla_x \ell(x + \eta, y)\right) = \frac{\sum_{i=1}^k u_i^2 \mathbb{1}_{\langle w, z_i\rangle + b \geq 0} \mathbb{1}_{\langle w, z_i + \eta_i\rangle + b \geq 0}}{\sqrt{\sum_{i=1}^k u_i^2 \mathbb{1}_{\langle w, z_i\rangle + b \geq 0} \sum_{i=1}^k u_i^2 \mathbb{1}_{\langle w, z_i + \eta_i\rangle + b \geq 0}}}. \tag{2.6}$$

Considering a single-filter CNN is also motivated by the fact that in the single-layer CNN introduced earlier, the norms of w_4 and its outcoming weights are much higher than for the rest of the filters (see Fig. 2.6), and thus the contribution of w_4 to the predictions and gradients of the network is the most significant. We observe that when an image x is convolved with the Laplace filter w_4, even a uniformly random noise η of small magnitude is able to significantly affect the output of $(x + \eta) * w_4$ (see Fig. 2.7). As a consequence, the ReLU activations of the network change their signs which directly affects the gradient alignment in Eq. (2.6). Namely, $x * w_4 + b_4$ has mostly negative values, and thus many values $\{\mathbb{1}_{\langle w_4, z_i\rangle + b_4}\}_{i=1}^k$ are equal to 0. On the other hand, nearly half of the values $\{\mathbb{1}_{\langle w_4, z_i + \eta_i\rangle + b_4}\}_{i=1}^k$ become 1, which significantly increases the denominator of Eq. (2.6), and thus makes the cosine close to 0. At the same time, the output of a regular filter w_1 shown in Fig. 2.7 is only slightly affected by the random noise η. For deep networks, however, we could not identify *particular* filters responsible for catastrophic overfitting, thus we consider next a more general solution.

2.6 Increasing gradient alignment improves fast adversarial training

Based on the importance of gradient alignment for successful FGSM training, we propose a regularizer, `GradAlign`, that aims at increasing gradient alignment and preventing catastrophic overfitting. The core idea of `GradAlign` is to maximize the gradient alignment (as defined in Eq. (2.5)) between the gradients at point x and at a randomly perturbed point $x + \eta$ inside the ℓ_∞-ball around x:

$$\Omega(x, y, \theta) \stackrel{\text{def}}{=} \mathbb{E}_{(x,y)\sim D,\ \eta\sim\mathcal{U}([-\varepsilon,\varepsilon]^d)} \left[1 - \cos\left(\nabla_x \ell(x, y; \theta), \nabla_x \ell(x + \eta, y; \theta)\right)\right]. \tag{2.7}$$

Crucially, `GradAlign` uses gradients at points x and $x + \eta$ which does not require an expensive iterative procedure unlike, e.g., the LLR method of Qin et al. (2019). Note that the regularizer depends only on the gradient direction and it is invariant to the gradient norm which contrasts it to the gradient penalties (Gu and Rigazio, 2015b; Hein and Andriushchenko, 2017b; Ross and Doshi-Velez, 2018; Simon-Gabriel et al., 2019) or CURE (Moosavi-Dezfooli et al., 2019b) (see the comparison in Appendix 2.11.5).

Experimental setup. We compare the following methods: standard FGSM AT, FGSM-RS AT with $\alpha = 1.25\varepsilon$ (Wong et al., 2020), FGSM AT + `GradAlign`, *AT for Free* with $m = 8$ (Shafahi et al., 2019), PGD-2 AT with 2-step PGD using $\alpha = \varepsilon/2$, and PGD-10 AT with 10-step PGD using $\alpha = 2\varepsilon/10$. We train these methods using PreAct ResNet-18 (He et al., 2016b) with ℓ_∞-radii $\varepsilon \in \{1/255, \dots, 16/255\}$ on CIFAR-10 for 30 epochs and $\varepsilon \in \{1/255, \dots, 12/255\}$ on SVHN for 15 epochs. The only exception is *AT for Free* (Shafahi et al., 2019) which we train for 96 epochs on CIFAR-10, and 45 epochs on SVHN which was necessary to get comparable results to the other methods. Unlike Qin et al. (2019) and Zhang et al. (2019a), with the training scheme of Wong et al. (2020) and $\alpha = \varepsilon/2$ we could successfully train a PGD-2 model with $\varepsilon = 8/255$ on CIFAR-10 with robustness better than that of their methods that use the same number of PGD steps (see Appendix 2.11.3). This also echoes the recent finding of Rice et al. (2020) that properly tuned multi-step PGD AT outperforms more recently published methods. As before, we evaluate robustness using PGD-50-10 with 50 iterations and 10 restarts using step size $\alpha = \varepsilon/4$ following Wong et al. (2020) for the same ε that was used for training. We train each model with 5 random seeds since the final robustness can have a large variance for high ε. Also, we remark that training with `GradAlign` leads on average to a 3× slowdown on an NVIDA V100 GPU compared to FGSM training which is due to the use of double backpropagation (see Etmann (2019) for a detailed analysis). We think that improving the runtime of `GradAlign` is possible, but we postpone it to future work. Additional implementation details are provided in Appendix 2.9. The code of our experiments is available at `https://github.com/tml-epfl/understanding-fast-adv-training`.

Results on CIFAR-10 and SVHN. We provide the main comparison in Fig. 2.8 and provide detailed numbers for specific values of ε in Appendix 2.11.3 which also includes an additional evaluation of our models with *AutoAttack* (Croce and Hein, 2020b). First, we

notice that all the methods perform almost equally well for small enough ε, i.e. $\varepsilon \leq 6/255$ on CIFAR-10 and $\varepsilon \leq 4/255$ on SVHN. However, the performance for larger ε varies a lot depending on the method due to catastrophic overfitting. Importantly, `GradAlign` *succesfully prevents catastrophic overfitting* in FGSM AT, thus allowing to successfully apply FGSM training also for larger ℓ_∞-perturbations and reduce the gap to PGD-10 training. In Appendix 2.11.4, we additionally show that FGSM + `GradAlign` does not suffer from catastrophic overfitting even for $\varepsilon \in \{24/255, 32/255\}$. At the same time, *not only* FGSM AT and FGSM-RS AT experience catastrophic overfitting, but also the recently proposed *AT for Free* and PGD-2, although at higher ε values than FGSM AT. We note that `GradAlign` is not only applicable to FGSM AT, but also to other methods that can also suffer from catastrophic overfitting. In particular, combining PGD-2 with `GradAlign` prevents catastrophic overfitting and leads to better robustness for $\varepsilon = 16/255$ on CIFAR-10 (see Appendix 2.11.3). Although performing early stopping can lead to non-trivial robustness, standard accuracy is often significantly sacrificed which *limits the usefulness of early stopping* as we show in Appendix 2.11.2. This is in contrast to training with `GradAlign` which leads to the same standard accuracy as PGD-10 AT.

Results on ImageNet. We also performed similar experiments on ImageNet in Appendix 2.11.3 to illustrate that `GradAlign` can be scaled to large-scale problems despite the slowdown. However, we observed that even for standard FGSM training using the training schedule of Wong et al. (2020), catastrophic overfitting *does not* occur for $\varepsilon \in \{2/255, 4/255\}$ considered in Shafahi et al. (2019); Wong et al. (2020), and thus there is no need to use `GradAlign` as its main role is to prevent catastrophic overfitting. We observe that for these ε values, the gradient alignment evolves similarly to that of PGD AT from the CIFAR-10 experiments shown in Fig. 2.4, i.e. it decreases gradually over epochs but *without* a sharp drop that would indicate catastrophic overfitting. For $\varepsilon = 6/255$, we observe that the gradient alignment and PGD accuracy for FGSM-RS drop very early in training (after 3 epochs), but not for FGSM or FGSM + `GradAlign` training. This contradicts our observations on CIFAR-10 and SVHN where we observed that FGSM-RS usually helps to postpone catastrophic overfitting to higher ε. However, it is computationally demanding to replicate the results on ImageNet over different random seeds as we did for CIFAR-10 and SVHN. Thus, we leave a more detailed investigation of catastrophic overfitting on ImageNet for future work.

Robust vs. catastrophic overfitting. Recently, Rice et al. (2020) brought up the importance of early stopping in adversarial training to mitigate the *robust overfitting* phenomenon that is characterized by a decreasing trend of test adversarial accuracy over training iterations. It is thus a natural question to ask whether robust and catastrophic overfitting are related, and whether `GradAlign` can be beneficial to mitigate robust overfitting. We observed that training FGSM + `GradAlign` for more than 30 epochs also leads to slightly worse robustness on the test set (see Appendix 2.11.4), thus suggesting that *catastrophic* and *robust overfitting* are two distinct phenomena that have to be addressed separately. As a sidenote, we also observe that FGSM training combined with `GradAlign` does not lead to catastrophic overfitting even when trained *up to 200 epochs*.

2.7 Conclusions and outlook

We observed that catastrophic overfitting is a fundamental problem not only for standard FGSM training, but for computationally efficient adversarial training in general. In particular, many recently proposed schemes such as FGSM AT enhanced by a random step or *AT for free* are also prone to catastrophic overfitting, and simply using early stopping leads to suboptimal models. Motivated by this, we explored the questions of *when* and *why* FGSM adversarial training works, and how to improve it by increasing the gradient alignment, and thus the quality of the solution of the inner maximization problem. Our proposed regularizer, `GradAlign`, prevents catastrophic overfitting and improves the robustness compared to other fast adversarial training methods reducing the gap to multi-step PGD training. However, `GradAlign` leads to an increased runtime due to the use of double backpropagation. We hope that the same effect of stabilizing the gradients under *random* noise can be achieved in future work with other regularization methods that do not rely on double backpropagation.

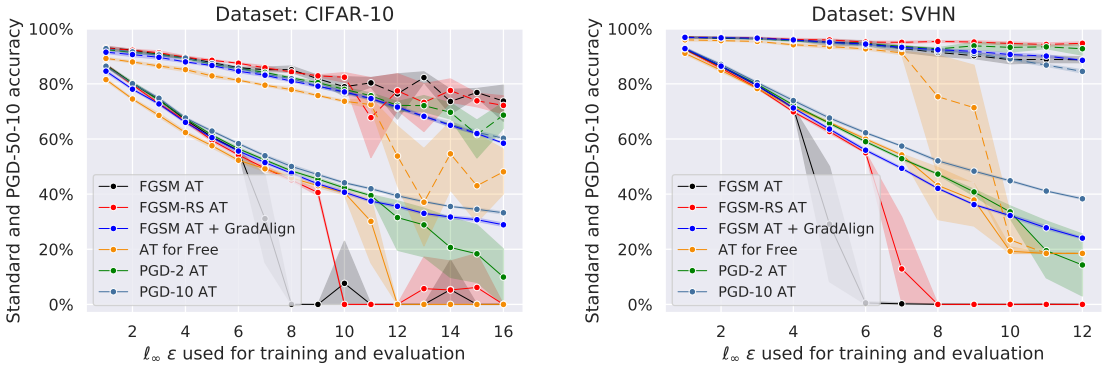


Figure 2.8: Accuracy (dashed line) and robustness (solid line) of different adversarial training (AT) methods on CIFAR-10 and SVHN with ResNet-18 trained and evaluated with different l_∞-radii. The results are obtained without early stopping, averaged over 5 random seeds used for training and reported with the standard deviation.

Appendix

2.8 Deferred proofs

In this section, we show the proofs omitted from Sec. 2.4 and Sec. 2.5.

2.8.1 Proof of Lemma 1

We state again Lemma 1 from Sec. 2.4 and present the proof.

Lemma 1. (Effect of the random step) *Let $\eta \sim \mathcal{U}([-\varepsilon, \varepsilon]^d)$ be a random starting point, and $\alpha \in [0, 2\varepsilon]$ be the step size of FGSM-RS defined in Eq. (2.3), then*

$$\mathbb{E}_\eta \left[\|\delta_{FGSM-RS}(\eta)\|_2\right] \leq \sqrt{\mathbb{E}_\eta \left[\|\delta_{FGSM-RS}(\eta)\|_2^2\right]} = \sqrt{d}\sqrt{-\frac{1}{6\varepsilon}\alpha^3 + \frac{1}{2}\alpha^2 + \frac{1}{3}\varepsilon^2}.$$

Proof. First, note that due to the Jensen's inequality, we can have a convenient upper bound which is easier to work with:

$$\mathbb{E}\left[\|\delta_{FGSM-RS}(\eta)\|_2\right] \leq \sqrt{\mathbb{E}\left[\|\delta_{FGSM-RS}(\eta)\|_2^2\right]}. \tag{2.8}$$

Therefore, we can focus on $\mathbb{E}\left[\|\delta_{FGSM-RS}\|_2^2\right]$ which can be computed analytically. Let us denote by $\nabla \stackrel{\text{def}}{=} \nabla_x \ell(x + \eta, y; \theta) \in \mathbb{R}^d$, we then obtain:

$$\begin{aligned}
\mathbb{E}_\eta \left[\|\delta_{FGSM-RS}\|_2^2\right] &= \mathbb{E}_\eta \left[\left\|\Pi_{[-\varepsilon,\varepsilon]}\left[\eta + \alpha \operatorname{sign}(\nabla)\right]\right\|_2^2\right] = \sum_{i=1}^d \mathbb{E}_{\eta_i}\left[\Pi_{[-\varepsilon,\varepsilon]}\left[\eta_i + \alpha \operatorname{sign}(\nabla_i)\right]^2\right] \\
&= d\,\mathbb{E}_{\eta_i}\left[\min\{\varepsilon, |\eta_i + \alpha \operatorname{sign}(\nabla_i)|\}^2\right] = d\,\mathbb{E}_{\eta_i}\left[\min\{\varepsilon^2, (\eta_i + \alpha \operatorname{sign}(\nabla_i))^2\}\right] \\
&= d\,\mathbb{E}_{r_i}\left[\mathbb{E}_{\eta_i}\left[\min\{\varepsilon^2, (\eta_i + \alpha \operatorname{sign}(\nabla_i))^2\} \mid \operatorname{sign}(\nabla_i) = r_i\right]\right],
\end{aligned}$$

where in the last step we use the law of total expectation by noting that $\operatorname{sign}(\nabla_i)$ is also a random variable since it depends on η_i.

We first consider the case when $\operatorname{sign}(\nabla_i) = 1$, then the inner conditional expectation is equal to:

$$\begin{aligned}
\int_{-\varepsilon}^{\varepsilon} \min\{\varepsilon^2, (\eta_i + \alpha)^2\}\frac{1}{2\varepsilon} d\eta_i &= \frac{1}{2\varepsilon}\int_{-\varepsilon+\alpha}^{\varepsilon+\alpha} \min\{\varepsilon^2, x^2\} dx \\
&= \frac{1}{2\varepsilon}\left(\int_{\varepsilon}^{\varepsilon+\alpha} \varepsilon^2 dx + \int_{-\varepsilon+\alpha}^{\varepsilon} x^2 dx\right) \\
&= -\frac{1}{6\varepsilon}\alpha^3 + \frac{1}{2}\alpha^2 + \frac{1}{3}\varepsilon^2.
\end{aligned}$$

The case when $\text{sign}(\nabla_i) = -1$ leads to the same expression:

$$\int_{-\varepsilon}^{\varepsilon} \min\{\varepsilon^2, (\eta_i - \alpha)^2\} \frac{1}{2\varepsilon} d\eta_i = \frac{1}{2\varepsilon} \int_{-\varepsilon-\alpha}^{\varepsilon-\alpha} \min\{\varepsilon^2, x^2\} dx = -\frac{1}{6\varepsilon}\alpha^3 + \frac{1}{2}\alpha^2 + \frac{1}{3}\varepsilon^2.$$

Combining these two cases together with Eq. (2.8), we have that:

$$\mathbb{E}_\eta \left[\|\delta_{FGSM-RS}(\eta)\|_2\right] \leq \sqrt{\mathbb{E}\left[\|\delta_{FGSM-RS}(\eta)\|_2^2\right]} = \sqrt{d}\sqrt{-\frac{1}{6\varepsilon}\alpha^3 + \frac{1}{2}\alpha^2 + \frac{1}{3}\varepsilon^2}.$$

□

2.8.2 Proof and discussion of Lemma 2

We state again Lemma 2 from Sec. 2.5 and present the proof.

Lemma 2. (Gradient alignment at initialization) *Let $z \sim \mathcal{U}([0,1]^p)$ be an image patch for $p \geq 2$, $\eta \sim \mathcal{U}([-\varepsilon, \varepsilon]^d)$ a point inside the ℓ_∞-ball, the parameters of a single-layer CNN initialized i.i.d. as $w \sim \mathcal{N}(0, \sigma_w^2 I_p)$ for every column of W, $u \sim \mathcal{N}(0, \sigma_u^2 I_m)$ for every column of U, $b := 0$, then the gradient alignment is lower bounded by*

$$\lim_{k,m\to\infty} \cos\left(\nabla_x \ell(x,y), \nabla_x \ell(x+\eta, y)\right) \geq \max\left\{1 - \sqrt{2}\,\mathbb{E}_{w,z}\left[e^{-\frac{1}{\varepsilon^2}\langle w/\|w\|_2, z\rangle^2}\right]^{1/2}, 0.5\right\}.$$

Proof. For k and m large enough, the law of large number ensures that an empirical mean of i.i.d. random variables can be approximated by its expectation with respect to random variables z, η, w, u. This leads to

$$\begin{aligned}
&\lim_{k,m\to\infty} \cos\left(\nabla_x \ell(x,y), \nabla_x \ell(x+\eta, y)\right) \\
&= \lim_{k,m\to\infty} \frac{\sum_{r=1}^{m}\sum_{l=1}^{m}\sum_{i=1}^{k} \langle w_r, w_l\rangle u_{ri}u_{li} \mathbb{1}_{\langle w_r, z_i\rangle \geq 0} \mathbb{1}_{\langle w_l, z_i+\eta_i\rangle \geq 0}}{\sqrt{\sum_{r=1}^{m}\sum_{l=1}^{m}\sum_{i=1}^{k} \langle w_r, w_l\rangle u_{ri}u_{li} \mathbb{1}_{\langle w_r, z_i\rangle \geq 0} \mathbb{1}_{\langle w_l, z_i\rangle \geq 0}} \sqrt{\sum_{r=1}^{m}\sum_{l=1}^{m}\sum_{i=1}^{k} \langle w_r, w_l\rangle u_{ri}u_{li} \mathbb{1}_{\langle w_r, z_i+\eta_i\rangle \geq 0} \mathbb{1}_{\langle w_l, z_i+\eta_i\rangle \geq 0}}} \\
&= \lim_{k,m\to\infty} \frac{\frac{1}{km}\sum_{r=1}^{m}\sum_{i=1}^{k} \|w_r\|_2^2 u_{ri}^2 \mathbb{1}_{\langle w_r, z_i\rangle \geq 0} \mathbb{1}_{\langle w_r, z_i+\eta_i\rangle \geq 0}}{\sqrt{\frac{1}{km}\sum_{r=1}^{m}\sum_{i=1}^{k} \|w_r\|_2^2 u_{ri}^2 \mathbb{1}_{\langle w_r, z_i\rangle \geq 0} \mathbb{1}_{\langle w_r, z_i\rangle \geq 0}} \sqrt{\frac{1}{km}\sum_{r=1}^{m}\sum_{i=1}^{k} \|w_r\|_2^2 u_{ri}^2 \mathbb{1}_{\langle w_r, z_i+\eta_i\rangle \geq 0} \mathbb{1}_{\langle w_r, z_i+\eta_i\rangle \geq 0}}} \\
&= \frac{\mathbb{E}_{w,u,\eta,z}\left[\|w\|_2^2 u^2 \mathbb{1}_{\langle w,z\rangle \geq 0} \mathbb{1}_{\langle w, z+\eta\rangle \geq 0}\right]}{\sqrt{\mathbb{E}_{w,u,z}\left[\|w\|_2^2 u^2 \mathbb{1}_{\langle w,z\rangle \geq 0}\right]}\sqrt{\mathbb{E}_{w,u,\eta,z}\left[\|w\|_2^2 u^2 \mathbb{1}_{\langle w,z+\eta\rangle \geq 0}\right]}} \\
&= \frac{\mathbb{E}_{w,z,\eta}\left[\|w\|_2^2 \mathbb{1}_{\langle w,z\rangle \geq 0} \mathbb{1}_{\langle w, z+\eta\rangle \geq 0}\right]}{\sqrt{\mathbb{E}_{w,z}\left[\|w\|_2^2 \mathbb{1}_{\langle w,z\rangle \geq 0}\right]}\sqrt{\mathbb{E}_{w,z,\eta}\left[\|w\|_2^2 \mathbb{1}_{\langle w,z+\eta\rangle \geq 0}\right]}}.
\end{aligned} \tag{2.9}$$

We directly compute for the denominator:

$$\mathbb{E}_{w,z}[\|w\|_2^2 \mathbb{1}_{\langle w,z\rangle \geq 0}] = \mathbb{E}_{w,\eta,z}[\|w\|_2^2 \mathbb{1}_{\langle w,z+\eta\rangle \geq 0}] = 0.5p\sigma_w^2.$$

For the numerator, by bounding $\mathbb{P}_\eta \left[\langle w, \eta \rangle \geq \langle w, z \rangle\right] \leq e^{-\frac{\langle z, w \rangle^2}{2\varepsilon^2 \|w\|_2^2}}$ via the Hoeffding's inequality, we obtain

$$
\begin{aligned}
\mathbb{E}_{u,w,z,\eta}\left[\|w\|_2^2\,\mathbb{1}_{\langle w,z\rangle\geq 0}\mathbb{1}_{\langle w,z+\eta\rangle\geq 0}\right] &= \mathbb{E}_{w,z,\eta}\left[\|w\|_2^2\,\mathbb{1}_{\langle w,z\rangle\geq 0}\mathbb{1}_{\langle w,z+\eta\rangle\geq 0}\right] \\
&= \mathbb{E}_{w,z}\left[\|w\|_2^2\,\mathbb{1}_{\langle w,z\rangle\geq 0}\mathbb{P}_\eta\left(\langle w,z+\eta\rangle\geq 0\right)\right] \\
&= \mathbb{E}_{w,z}\left[\|w\|_2^2\,\mathbb{1}_{\langle w,z\rangle\geq 0}\mathbb{P}_\eta\left(\langle w,\eta\rangle\geq -\langle w,z\rangle\right)\right] \\
&= \mathbb{E}_{w,z}\left[\|w\|_2^2\,\mathbb{1}_{\langle w,z\rangle\geq 0}\mathbb{P}_\eta\left(\langle w,\eta\rangle\leq \langle w,z\rangle\right)\right] \\
&= \mathbb{E}_{w,z}\left[\|w\|_2^2\,\mathbb{1}_{\langle w,z\rangle\geq 0}\left(1-\mathbb{P}_\eta\left(\langle w,\eta\rangle\geq \langle w,z\rangle\right)\right)\right] \\
&\geq \mathbb{E}_{w,z}\left[\|w\|_2^2\,\mathbb{1}_{\langle w,z\rangle\geq 0}\left(1-e^{-\frac{\langle w,z\rangle^2}{2\varepsilon^2\|w\|_2^2}}\right)\right] \\
&= \mathbb{E}_{w,z}\left[\|w\|_2^2\,\mathbb{1}_{\langle w,z\rangle\geq 0}\right] - \mathbb{E}_{w,z}\left[\|w\|_2^2\,\mathbb{1}_{\langle w,z\rangle\geq 0}\,e^{-\frac{\langle w,z\rangle^2}{2\varepsilon^2\|w\|_2^2}}\right] \\
&= 0.5p\sigma_w^2 - 0.5\,\mathbb{E}_{w,z}\left[\|w\|_2^2\,e^{-\frac{\langle w,z\rangle^2}{2\varepsilon^2\|w\|_2^2}}\right] \\
&\geq 0.5p\sigma_w^2 - 0.5\,\mathbb{E}_w\left[\|w\|_2^4\right]^{1/2}\mathbb{E}_{w,z}\left[e^{-\frac{\langle w,z\rangle^2}{\varepsilon^2\|w\|_2^2}}\right]^{1/2} \\
&= 0.5p\sigma_w^2 - 0.5\sigma_w^2\sqrt{p^2+2p}\,\mathbb{E}_{w,z}\left[e^{-\frac{\langle w,z\rangle^2}{\varepsilon^2\|w\|_2^2}}\right]^{1/2},
\end{aligned}
$$

where the last inequality is obtained via the Cauchy-Schwarz inequality. On the other hand, we have:

$$
\begin{aligned}
\mathbb{E}_{u,w,z,\eta}\left[\|w\|_2^2\,\mathbb{1}_{\langle w,z\rangle\geq 0}\mathbb{1}_{\langle w,z+\eta\rangle\geq 0}\right] &= \mathbb{E}_{w,z}\left[\|w\|_2^2\,\mathbb{1}_{\langle w,z\rangle\geq 0}\mathbb{P}_\eta\left(\langle w,\eta\rangle\leq \langle w,z\rangle\right)\right] \\
&\geq \mathbb{E}_{w,z}\left[\|w\|_2^2\,\mathbb{1}_{\langle w,z\rangle\geq 0}0.5\right] = 0.25p\sigma_w^2.
\end{aligned}
$$

Now we combine both lower bounds together to establish a lower bound on Eq. (2.9):

$$
\begin{aligned}
&\frac{\mathbb{E}_{w,z,\eta}\left[\|w\|_2^2\,\mathbb{1}_{\langle w,z\rangle\geq 0}\mathbb{1}_{\langle w,z+\eta\rangle\geq 0}\right]}{\sqrt{\mathbb{E}_{w,z}\left[\|w\|_2^2\,\mathbb{1}_{\langle w,z\rangle\geq 0}\right]}\sqrt{\mathbb{E}_{w,z,\eta}\left[\|w\|_2^2\,\mathbb{1}_{\langle w,z+\eta\rangle\geq 0}\right]}} \\
\geq &\frac{\max\left\{0.5p\sigma_w^2 - 0.5\sigma_w^2\sqrt{p^2+2p}\,\mathbb{E}_{w,z}\left[e^{-\frac{\langle w,z\rangle^2}{\varepsilon^2\|w\|_2^2}}\right]^{1/2}, 0.25p\sigma_w^2\right\}}{0.5p\sigma_w^2} \\
= &\max\left\{1-\sqrt{1+\frac{2}{p}}\,\mathbb{E}_{w,z}\left[e^{-\frac{\langle w/\|w\|_2,z\rangle^2}{\varepsilon^2}}\right]^{1/2}, 0.5\right\} \\
\geq &\max\left\{1-\sqrt{2}\,\mathbb{E}_{w,z}\left[e^{-\frac{1}{\varepsilon^2}\langle w/\|w\|_2,z\rangle^2}\right]^{1/2}, 0.5\right\},
\end{aligned}
\tag{2.10}
$$

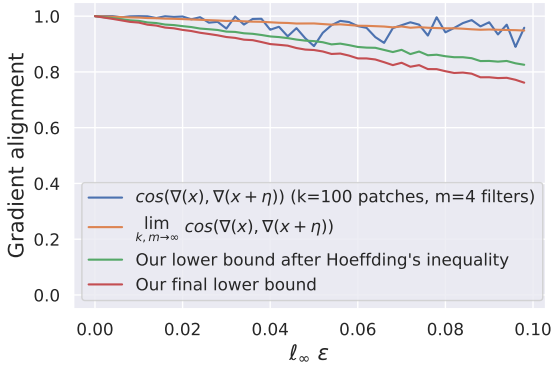


Figure 2.9: Visualization of the key quantities involved in Lemma 2.

where in the last step we used that $p \geq 2$. □

The main purpose of obtaining the lower bound in Lemma 2 was to get an expression that can give us an insight into the key quantities which gradient alignment at initialization depends on. Considering the limiting case $k, m \to \infty$ was necessary to obtain a ratio of expectations that allowed us to derive a simpler expression. Finally, we lower bounded the gradient alignment from Eq. (2.9) using the Hoeffding's and Cauchy-Schwarz inequalities and used $p \geq 2$ to obtain a dimension-independent constant in front of the expectation in Eq. (2.10). Now we would like to provide a better understanding about the key quantities involved in the lemma and to assess the tightness of the derived lower bound. For this purpose, in Fig. 2.9 we plot:

- $\cos\left(\nabla_x \ell(x, y), \nabla_x \ell(x + \eta, y)\right)$ for $k = 100$ patches and $m = 4$ filters (which resembles the setting of the 4-filter CNN on CIFAR-10). We note that it is a random variable since it is a function of random variables x, η, W, U.
- $\lim_{k,m\to\infty} \cos\left(\nabla_x \ell(x, y), \nabla_x \ell(x + \eta, y)\right)$ evaluated via Eq. (2.9).
- Our first lower bound $\max\left\{1 - \frac{1}{p\sigma_w^2}\, \mathbb{E}_{w,z}\left[\|w\|_2^2\, e^{-\frac{1}{2\varepsilon^2}\langle w/\|w\|_2, z\rangle^2}\right], 0.5\right\}$ obtained via Hoeffding's inequality.
- Our final lower bound $\max\left\{1 - \sqrt{2}\, \mathbb{E}_{w,z}\left[e^{-\frac{1}{\varepsilon^2}\langle w/\|w\|_2, z\rangle^2}\right]^{1/2}, 0.5\right\}$.

For the last three quantities we approximate the expectations by Monte-Carlo sampling by using 1,000 samples. For all the quantities we use patches of size $p = 3 \times 3 \times 3 = 27$ as in our CIFAR-10 experiments. We plot gradient alignment values for $\varepsilon \in [0, 0.1]$ since we are interested in small ℓ_∞-perturbations such as, e.g., $\varepsilon = 8/255 \approx 0.03$ which is a typical value used for CIFAR-10 (Madry et al., 2018). First, we can observe that all the four quantities have very high values in $[0.7, 1.0]$ for $\varepsilon \in [0, 0.1]$ which is in contrast to the gradient alignment value of 0.12 that we observe after catastrophic overfitting for

$\varepsilon = {}^{10}/_{255} \approx 0.04$. Next, we observe that $\cos\left(\nabla_x \ell(x, y), \nabla_x \ell(x + \eta, y)\right)$ has some noticeable variance for the chosen parameters $k = 100$ patches and $m = 4$ filters. However, this variance is significantly reduced when we increase the parameters k and m, especially when considering the limiting case $k, m \to \infty$. Finally, we observe that both lower bounds on $\lim_{k,m\to\infty} \cos\left(\nabla_x \ell(x, y), \nabla_x \ell(x + \eta, y)\right)$ that we derived are empirically tight enough to properly capture the behaviour of gradient alignment for small ε. However, we choose to report the last one in the lemma since it is slightly more concise than the one obtained via Hoeffding's inequality.

2.9 Experimental details

We list detailed evaluation and training details below.

Evaluation. Throughout the paper, we use PGD-50-10 for evaluation of adversarial accuracy which stands for the PGD attack with 50 iterations and 10 random restarts following Wong et al. (2020). We use the step size $\alpha = {}^{\varepsilon}/_{4}$. The choice of this attack is motivated by the fact that in both public benchmarks of Madry et al. (2018) on MNIST and CIFAR-10, the adversarial accuracy of PGD-100-50 and PGD-20-10 respectively is only 2% away from the best entries.

Although we train our models using half precision (Micikevicius et al., 2018), we always perform robustness evaluation using single precision since evaluation with half precision can sometimes overestimate the robustness of the model due to limited numerical precision in the calculation of the gradients.

We perform evaluation of standard accuracy using full test sets, but we evaluate adversarial accuracy using 1,000 random points on each dataset.

Training details for ResNet-18. We use the implementation code of Wong et al. (2020) with the only difference that we do not use image normalization and gradient clipping on CIFAR-10 and SVHN since we found that they have no significant influence on the final results. We use cyclic learning rates and half-precision training following Wong et al. (2020). We do not use random initialization for PGD during adversarial training as we did not find that it leads to any improvements on the considered datasets (see the justifications in Sec. 2.11.1 below). We perform early stopping based on the PGD accuracy on the training set following Wong et al. (2020). We observed that such a simple model selection scheme can successfully select a model before catastrophic overfitting that has non-trivial robustness.

On CIFAR-10, we train all the models for 30 epochs with the maximum learning rate 0.3 except *AT for free* (Shafahi et al., 2019) which we train for 96 epochs with the maximum learning rate 0.04 using $m = 8$ minibatch replays to get comparable results to the other methods.

On SVHN, we train all the models for 15 epochs with the maximum learning rate 0.05 except *AT for free* (Shafahi et al., 2019) which we train for 45 epochs with the maximum learning rate 0.01 using $m = 8$ minibatch replays. Moreover, in order to prevent convergence to a constant classifier on SVHN, we linearly increase the perturbation radius from 0 to ε during the first 5 epochs for all methods.

For PGD-2 AT we use for training a 2-step PGD attack with step size $\alpha = \varepsilon/2$, and for PGD-10 AT we use for training a 10-step PGD attack with $\alpha = 2\varepsilon/10$.

For Fig. 2.1 and Fig. 2.8 we used the `GradAlign` λ values obtained via a linear interpolation on the logarithmic scale between the best λ values that we found for $\varepsilon = 8$ and $\varepsilon = 16$ on the test sets. We perform the interpolation on the logarithmic scale since the values of λ are non-negative, a usual linear interpolation would lead to negative values of λ. The resulting λ values for $\varepsilon \in \{1, \dots, 16\}$ are given in Table 2.2. We note that at the end we do not report the results with $\varepsilon > 12$ for SVHN since many models have trivial robustness close to that of a constant classifier. For the PGD-2 + `GradAlign` experiments reported

Table 2.2: `GradAlign` λ values used for the experiments on CIFAR-10 and SVHN. These values are obtained via a linear interpolation on the logarithmic scale between successful λ values at $\varepsilon = 8$ and $\varepsilon = 16$.

ε (/255)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$\lambda_{CIFAR-10}$	0.03	0.04	0.05	0.06	0.08	0.11	0.15	0.20	0.27	0.36	0.47	0.63	0.84	1.12	1.50	2.00
λ_{SVHN}	1.66	1.76	1.86	1.98	2.10	2.22	2.36	2.50	2.65	2.81	2.98	3.16	3.35	3.56	3.77	4.00

below in Table 2.4 and Table 2.5, we use $\lambda = 0.1$ for the CIFAR-10 and $\lambda = 0.5$ for SVHN experiments.

Training details for the single-layer CNN. The single-layer CNN that we study in Sec. 2.5 has 4 convolutional filters, each of them of size 3×3. After the convolution we apply ReLU activation, and then we directly have a fully-connected layer, i.e. we do not use any pooling layer. For training we use the ADAM optimizer with learning rate 0.003 for 30 epochs using the same cyclical learning rate schedule.

ImageNet experiments. We use ResNet-50 following the training scheme of Wong et al. (2020) which includes 3 training stages on different image resolution. For `GradAlign`, we slightly reduce the batch size on the second and third stages from 224 and 128 to 180 and 100 respectively in order to reduce the memory consumption. For all $\varepsilon \in \{2, 4, 6\}$, we train FGSM models with `GradAlign` using $\lambda \in \{0.01, 0.1\}$. The final λ we report are $\lambda \in \{0.01, 0.01, 0.1\}$ for $\varepsilon \in \{2, 4, 6\}$ respectively.

Computing infrastructure. We perform all our experiments on NVIDIA V100 GPUs with 32GB of memory.

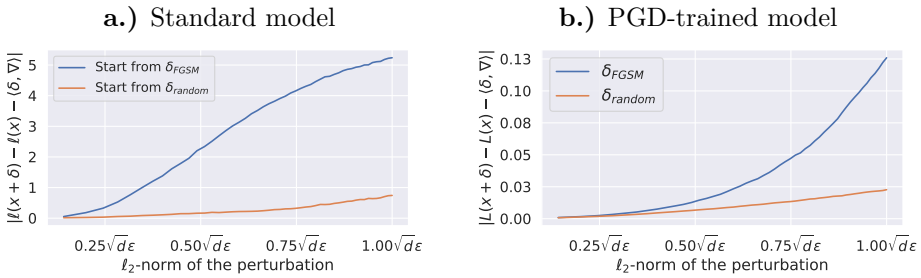


Figure 2.10: The quality of the linear approximation of $\ell(x+\delta)$ for δ with different ℓ_2-norm for $\|\delta\|_\infty$ fixed to ε for a standard and PGD-trained ResNet-18 on CIFAR-10.

2.10 Supporting experiments and visualizations for Sec. 2.4 and Sec. 2.5

We describe here supporting experiments and visualizations related to Sec. 2.4 and Sec. 2.5.

2.10.1 Quality of the linear approximation for ReLU networks

For the loss function ℓ of a ReLU-network, we compute empirically the quality of the linear approximation defined as

$$|\ell(x+\delta) - \ell(x) - \langle \delta, \nabla_x \ell(x) \rangle|,$$

where the dependency of the loss ℓ on the label y and parameters θ are omitted for clarity. Then we perform the following experiment: we take a perturbation $\delta \in \{-\varepsilon, \varepsilon\}^d$, and then zero out different fractions of its coordinates, which leads to perturbations with a fixed $\|\delta\|_\infty = \varepsilon$, but with different $\|\delta\|_2 \in [0, \sqrt{d}\varepsilon]$. As the starting δ we choose two types of perturbations: δ_{FGSM} generated by FGSM and δ_{random} sampled uniformly from the corners of the ℓ_∞-ball. We plot the results in Fig. 2.10 on CIFAR-10 for $\varepsilon = 8/255$ averaged over 512 test points, and conclude that for both δ_{FGSM} and δ_{random} the validity of the linear approximation crucially depends on $\|\delta\|_2$ even when $\|\delta\|_\infty$ is fixed. The phenomenon is even more pronounced for FGSM perturbations as the linearization error is much higher there. Moreover, this observation is consistent across both standardly and adversarially trained ResNet-18 models.

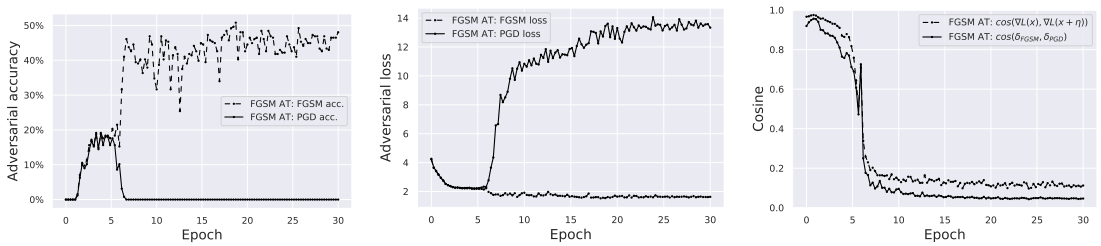


Figure 2.11: Visualization of the training process of an FGSM trained CNN with 4 filters with $\varepsilon = 10/255$. We can observe catastrophic overfitting around epoch 6.

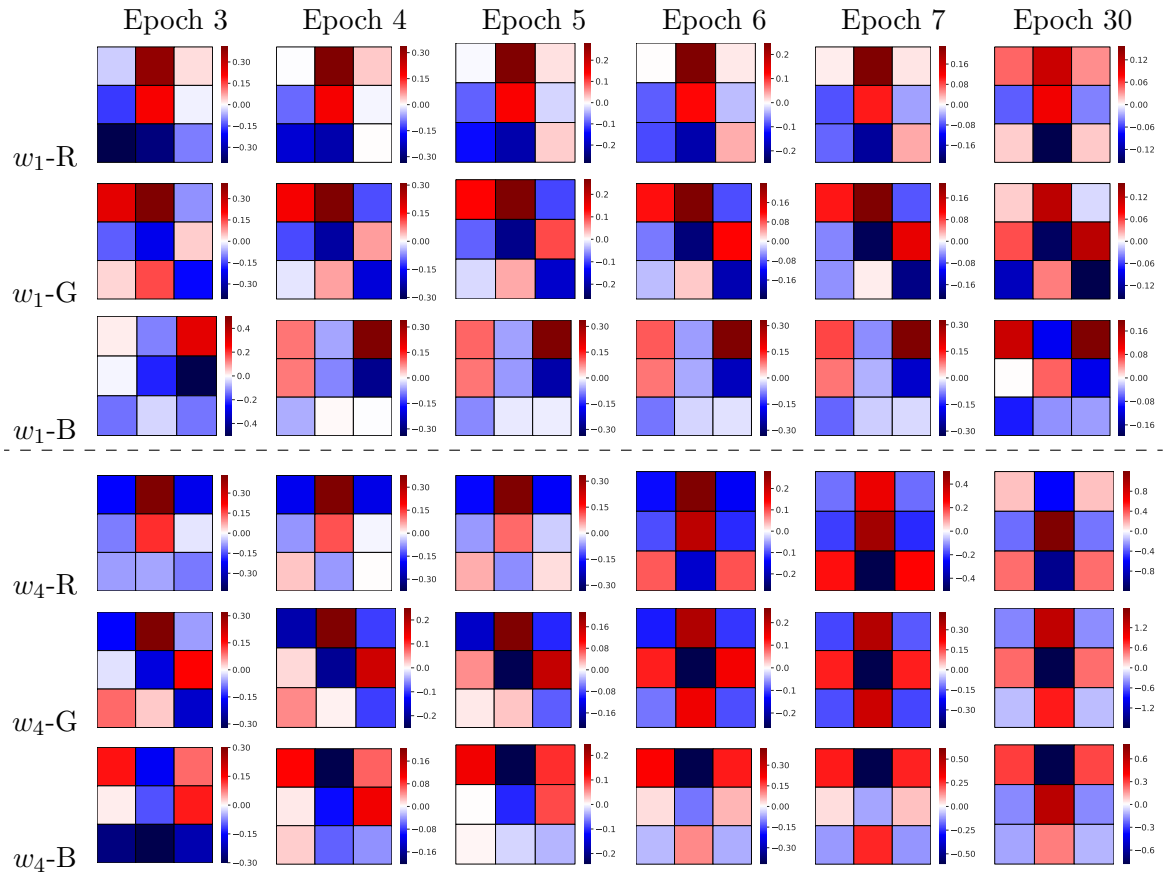


Figure 2.12: Evolution of the regular filter w_1 and filter w_4 that leads to catastrophic overfitting. We plot red (R), green (G), and blue (B) channels of the filters. We can observe that in R and G channels, w_4 has learned a Laplace filter which is very sensitive to noise.

2.10.2 Catastrophic overfitting in a single-layer CNN

We describe here complementary figures to Sec. 2.5 which are related to the single-layer CNN.

Training curves. In Fig. 2.11, we show the evolution of the FGSM/PGD accuracy, FGSM/PGD loss, and gradient alignment together with $\cos(\delta_{FGSM}, \delta_{PGD})$. We observe that catastrophic overfitting occurs around epoch 6 and that its pattern is the same as for the deep ResNet which was illustrated in Fig. 2.4. Namely, we see that concurrently the following changes occur around epoch 6: (a) there is a sudden drop of PGD accuracy with an increase in FGSM accuracy, (b) the PGD loss grows by an order of magnitude while the FGSM loss decreases, (c) both gradient alignment and $\cos(\delta_{FGSM}, \delta_{PGD})$ significantly decrease. Throughout all our experiments we observe a very high correlation between $\cos(\delta_{FGSM}, \delta_{PGD})$ and gradient alignment. This motivates our proposed regularizer `GradAlign` which relies on the cosine between $\nabla_x \ell(x, y; \theta)$ and $\nabla_x \ell(x + \eta, y; \theta)$, where η is a *random* point. In this way, we avoid using an iterative procedure inside the regularizer unlike, for example, the approach of Qin et al. (2019).

Additional filters. In Fig. 2.12, we show the evolution of the regular filter w_1 and filter w_4 that leads to catastrophic overfitting for the three input channels (red, green, blue). We can observe that in the red and green channels, w_4 has learned a Laplace filter which is very sensitive to noise. Moreover, w_4 significantly increases in magnitude after catastrophic overfitting contrary to w_1 whose magnitude only decreases (see the colorbar values in Fig. 2.12 and the plots in Fig. 2.5).

Additional feature maps. In Fig. 2.13, we show additional feature maps for images with and without uniform random noise $\eta \sim \mathcal{U}([-10/255, 10/255]^d)$. These figures complement Fig. 2.7 shown in the main part. We clearly see that only the last filter w_4 is sensitive to the noise since the feature maps change dramatically. At the same time, other filters w_1, w_2, w_3 are only slightly affected by the addition of the noise. We also show the input gradients in the last column which illustrate that after adding the noise the gradients change drammatically which leads to small gradient alignment and, in turn, to the failure of FGSM as the solution of the inner maximization problem.

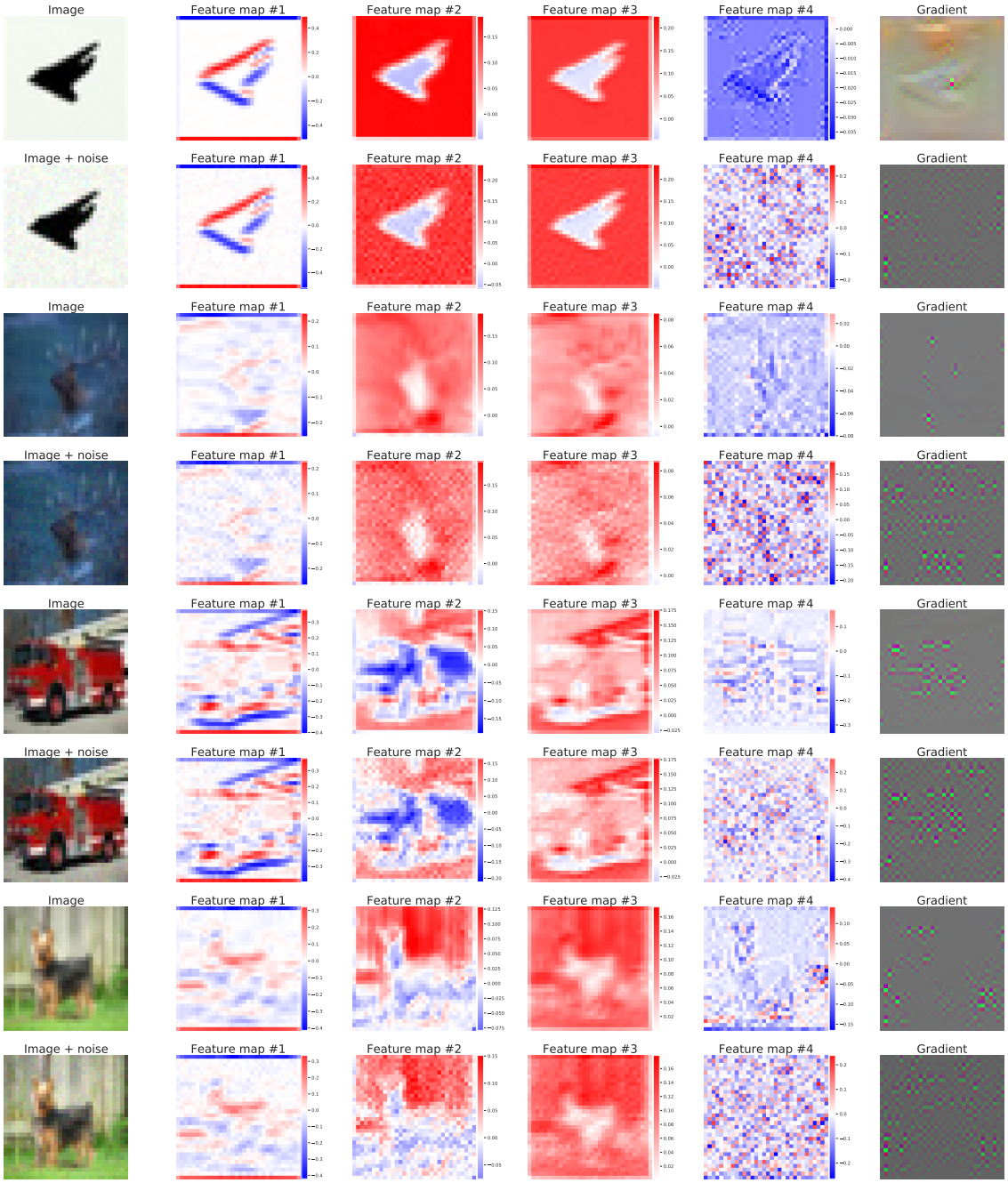


Figure 2.13: Input images, feature maps, and gradients of the single-layer CNN trained on CIFAR-10 at the end of training (after catastrophic overfitting). *Odd row*: original images. *Even row*: original image plus random noise $\mathcal{U}([-10/255, 10/255]^d)$. We observe that only the last filter w_4 is highly sensitive to the small uniform noise since the feature maps change dramatically.

2.11 Additional experiments for different adversarial training schemes

In this section, we describe additional experiments related to `GradAlign` that complement the results shown in Sec. 2.6.

2.11.1 Stronger PGD-2 baseline

As mentioned in Sec. 2.6, the PGD-2 training baseline that we report outperforms other similar baselines reported in the literature (Zhang et al., 2019a; Qin et al., 2019). Here we elaborate what are likely to be the most important sources of difference. First, we follow the cyclical learning rate schedule of Wong et al. (2020) which can work as implicit early stopping and thus can help to prevent catastrophic overfitting observed for PGD-2 in Qin et al. (2019). Another source of difference is that Qin et al. (2019) use the ADAM optimizer while we stick to the standard PGD updates using the sign of the gradient (Madry et al., 2018).

The second important factor is a proper step size selection. While Zhang et al. (2019a) do not observe catastrophic overfitting, their PGD-3 baseline achieves only 32.51% adversarial accuracy compared to the 48.43% for our PGD-2 baseline evaluated with a stronger attack (PGD-50-10 instead of PGD-20-1). One potential explanation for this difference lies in the step size selection, where for PGD-2 we use $\alpha = \varepsilon/2$. Related to the step size selection, we also found that using random initialization in PGD (we will refer to as PGD-k-RS) as suggested in Madry et al. (2018) requires a larger step size α. We show the results in Table 2.3 where we can see that PGD-2-RS AT with $\alpha = \varepsilon/2$ achieves suboptimal robustness compared to $\alpha = \varepsilon$ used for training. However, we consistently observed that PGD-2 AT with $\alpha = \varepsilon/2$ and *no random step* performs best. Thus, we use the latter as our PGD-2 baseline throughout the paper, thus always starting PGD-2 from the original point, without using any random step.

Table 2.3: Robustness of different PGD-2 schemes for $\varepsilon = 8/255$ on CIFAR-10 for ResNet-18. The results are averaged over 5 random seeds used for training.

Model	PGD-2-RS AT, $\alpha = \varepsilon/2$	PGD-2-RS AT, $\alpha = \varepsilon$	PGD-2 AT, $\alpha = \varepsilon/2$
PGD-50-10 accuracy	45.06±0.44%	48.07±0.52%	48.43±0.40%

2.11.2 Results with early stopping

We complement the results presented in Fig. 2.8 *without early stopping* with the results *with early stopping* which we show in Fig. 2.14. For CIFAR-10, we observe that FGSM + `GradAlign` leads to a good robustness and accuracy outperforming FGSM AT and FGSM-RS AT and performing similarly to PGD-2 and slightly improving for larger ε close to $16/255$. For SVHN, `GradAlign` leads to better robustness than other FGSM-based

methods. We also observe that for large ε on both CIFAR-10 and SVHN, *AT for Free* performs similarly to FGSM-based methods. Moreover, for $\varepsilon \geq 10/255$ on SVHN, *AT for Free* converges to a constant classifier.

On both CIFAR-10 and SVHN, we can see that although early stopping can lead to non-trivial robustness, standard accuracy is often significantly sacrificed which limits the usefulness of this technique. This is in contrast to training with `GradAlign` which leads to the same standard accuracy as PGD-10 training.

2.11.3 Results for specific ℓ_∞-radii

Here we report results from Fig. 2.8 for specific ℓ_∞-radii which are most often studied in the literature.

CIFAR-10 results. We report robustness and accuracy in Table 2.4 for CIFAR-10 without using early stopping where we can clearly see which methods lead to catastrophic overfitting and thus suboptimal robustness. We compare the same methods as in Fig. 2.8, and additionally we report the results for $\varepsilon = 8/255$ of the CURE (Moosavi-Dezfooli

Table 2.4: Robustness and accuracy of different robust training methods on **CIFAR-10**. We report results without early stopping for ResNet-18 unless specified otherwise in parentheses. The results of all the methods reported in Fig. 2.8 are shown here with the standard deviation and averaged over 5 random seeds used for training.

Model	**Accuracy**		**Attack**
	Standard	Adversarial	
	$\varepsilon = 8/255$		
Standard	94.03%	0.00%	PGD-50-10
CURE	81.20%	36.30%	PGD-20-1
YOPO-3-5	82.14%	38.18%	PGD-20-1
YOPO-5-3	83.99%	44.72%	PGD-20-1
LLR-2 (Wide-ResNet-28-8)	90.46%	44.50%	MultiTargeted
FGSM	85.16±1.3%	0.02±0.04%	PGD-50-10
FGSM-RS	84.32±0.08%	45.10±0.56%	PGD-50-10
FGSM + `GradAlign`	81.00±0.37%	**47.58±0.24%**	PGD-50-10
AT for Free ($m = 8$)	77.92±0.65%	45.90±0.98%	PGD-50-10
PGD-2 ($\alpha = 4/255$)	82.15±0.48%	48.43±0.40%	PGD-50-10
PGD-2 ($\alpha = 4/255$) + `GradAlign`	81.16±0.39%	47.76±0.77%	PGD-50-10
PGD-10 ($\alpha = 2\varepsilon/10$)	81.88±0.37%	**50.04±0.79%**	PGD-50-10
	$\varepsilon = 16/255$		
FGSM	73.76±7.4%	0.00±0.00%	PGD-50-10
FGSM-RS	72.18±3.7%	0.00±0.00%	PGD-50-10
FGSM + `GradAlign`	58.46±0.22%	**28.88±0.70%**	PGD-50-10
AT for Free ($m = 8$)	48.10±9.83%	0.00±0.00%	PGD-50-10
PGD-2 ($\alpha = \varepsilon/2$)	68.65±5.83%	9.92±14.00%	PGD-50-10
PGD-2 ($\alpha = \varepsilon/2$) + `GradAlign`	61.38±0.71%	29.80±0.42%	PGD-50-10
PGD-10 ($\alpha = 2\varepsilon/10$)	60.28±0.50%	**33.24±0.52%**	PGD-50-10

et al., 2019b), YOPO (Zhang et al., 2019a), and LLR (Qin et al., 2019) approaches. First, for $\varepsilon = 8/255$, we see that FGSM + `GradAlign` outperforms *AT for Free* and all methods that use FGSM training. Then, we also observe that the model trained with CURE (Moosavi-Dezfooli et al., 2019b) leads to robustness that is suboptimal compared to FGSM-RS AT evaluated with a stronger attack: 36.3% vs 45.1%. YOPO-3-5 and YOPO-5-3 (Zhang et al., 2019a) require 3 and 5 full steps of PGD respectively, thus they are much more expensive than FGSM-RS AT, and, however, they lead to worse adversarial accuracy: 38.18% and 44.72% vs 45.10%. Qin et al. (2019) report that LLR-2, i.e. their approach with 2 steps of PGD, achieves 44.50% adversarial accuracy with MultiTargeted attack (Gowal et al., 2019b) and 46.47% with their untargeted PGD attack which uses a different loss function compared to our PGD attack. These two evaluations are not directly comparable to other results in Table 2.4 since the attacks are different and moreover they use a larger network (Wide-ResNet-28-8) which usually leads to better results (Madry et al., 2018). However, we think that the gap of $3-4$% adversarial accuracy of MultiTargeted evaluation compared to that of our reported FGSM + `GradAlign` and PGD-2 methods (47.58% and 48.43% resp.) is still significant since the difference between MultiTargeted and a PGD attack with random restarts is observed to be small (e.g. around 1% between MultiTargeted and PGD-20-10 on the CIFAR-10 challenge of Madry et al. (2018)).

For $\varepsilon = 16/255$, none of the one-step methods work without early stopping except FGSM with `GradAlign`. We also evaluate PGD-2 + `GradAlign` and conclude that the benefit of combining the two comes when PGD-2 alone leads to catastrophic overfitting which occurs at $\varepsilon = 16/255$. For $\varepsilon = 8/255$, there is no benefit of combining the two approaches. This is consistent with our observation regarding catastrophic overfitting for FGSM (e.g. see Fig. 2.8 for small ε): if there is no catastrophic overfitting, there is no benefit of adding `GradAlign` to FGSM training.

To further ensure that FGSM + `GradAlign` models do not benefit from gradient masking (Papernot et al., 2017), we additionally compare the robustness of FGSM + `GradAlign` and FGSM-RS models obtained via *AutoAttack* (Croce and Hein, 2020b). We observe that *AutoAttack* proportionally reduces the adversarial accuracy of both models: for $\varepsilon = 8/255$, FGSM + `GradAlign` achieves 44.54±0.24% adversarial accuracy while FGSM-RS achieves 42.80±0.58%. This is consistent with the evaluation results of Croce and Hein (2020b) where they show that *AutoAttack* reduces adversarial accuracy for many models by 2%-3% for $\varepsilon = 8/255$ compared to the originally reported results based on the standard PGD attack (see Table 2 in Croce and Hein (2020b)). The same tendency is observed also for higher ε, e.g. for $\varepsilon = 16/255$ FGSM + `GradAlign` achieves 20.56±0.36% adversarial accuracy when evaluated with *AutoAttack*.

SVHN results. We report robustness and accuracy in Table 2.5 for SVHN without using early stopping. We can see that for both $\varepsilon = 8/255$ and $\varepsilon = 16/255$, `GradAlign` successfully prevents catastrophic overfitting in contrast to FGSM and FGSM-RS, although there is still a 5% gap to PGD-2 training for $\varepsilon = 8/255$. *AT for free* performs slightly better than

Table 2.5: Robustness and accuracy of different robust training methods on **SVHN**. We report results without early stopping for ResNet-18. All the results are reported with the standard deviation and averaged over 5 random seeds used for training.

Model	Accuracy	
	Standard	PGD-50-10
$\varepsilon = 8/255$		
Standard	96.00%	1.00%
FGSM	91.40±1.64%	0.04±0.05%
FGSM-RS	95.38±0.27%	0.00±0.00%
FGSM + `GradAlign`	92.36±0.47%	**42.08±0.25%**
AT for Free ($m = 8$)	75.34±28.4%	43.16±12.3%
PGD-2 ($\alpha = \varepsilon/2$)	92.68±0.45%	47.28±0.26%
PGD-2 + `GradAlign` ($\alpha = \varepsilon/2$)	92.46±0.35%	47.02±0.83%
PGD-10 ($\alpha = 2\varepsilon/10$)	91.92±0.40%	**52.08±0.49%**
$\varepsilon = 12/255$		
FGSM	88.74±1.25%	0.00±0.00%
FGSM-RS	94.70±0.66%	0.00±0.00%
FGSM + `GradAlign`	88.54±0.21%	**24.04±0.31%**
AT for Free ($m = 8$)	18.50±0.00%	18.50±0.00%
PGD-2 ($\alpha = \varepsilon/2$)	92.74±2.26%	14.30±13.34%
PGD-2 + `GradAlign` ($\alpha = \varepsilon/2$)	87.14±0.26%	31.26±0.24%
PGD-10 ($\alpha = 2\varepsilon/10$)	84.52±0.63%	**38.32±0.38%**

Table 2.6: Robustness and accuracy of different robust training methods on **ImageNet**. We report results without early stopping for ResNet-50.

Model	ℓ_∞-radius	Standard accuracy	PGD-50-10 accuracy
FGSM	2/255	61.7%	42.1%
FGSM-RS	2/255	59.3%	41.1%
FGSM + `GradAlign`	2/255	61.8%	41.4%
FGSM	4/255	56.9%	30.6%
FGSM-RS	4/255	55.3%	27.8%
FGSM + `GradAlign`	4/255	57.8%	30.5%
FGSM	6/255	51.5%	20.6%
FGSM-RS	6/255	36.6%	0.1%
FGSM + `GradAlign`	6/255	51.5%	20.3%

FGSM + `GradAlign` for $\varepsilon = 8/255$, but it already starts to show a high variance in the robustness and accuracy depending on the random seed. For $\varepsilon = 12/255$, all the 5 models of *AT for free* converge to a constant classifier.

Combining PGD-2 with `GradAlign` does not lead to improved results for $\varepsilon = 8/255$ since there is no catastrophic overfitting for PGD-2. However, for $\varepsilon = 12/255$, we can clearly see that PGD-2 + `GradAlign` leads to better results than PGD-2 achieving $31.26 \pm 0.24\%$ instead of $14.30 \pm 13.34\%$ adversarial accuracy.

ImageNet results. We also perform similar experiments on ImageNet in Table 2.6. We observe that even for standard FGSM training, catastrophic overfitting *does not* occur for

$\varepsilon \in \{2/255, 4/255\}$ considered in Shafahi et al. (2019); Wong et al. (2020), and thus there is no additional benefit from using `GradAlign` since its main role is to prevent catastrophic overfitting. We report the results of FGSM + `GradAlign` for completeness to show that `GradAlign` can be applied on the ImageNet scale, although it leads to approximately 3× slowdown on ImageNet compared to standard FGSM training.

For $\varepsilon = 6/255$, we observe that catastrophic overfitting occurs for FGSM-RS very early in training (around epoch 3), but not for FGSM or FGSM + `GradAlign` training. This contradicts our observations on CIFAR-10 and SVHN where we observed that FGSM-RS usually helps to postpone catastrophic overfitting to higher ε. However, it is computationally demanding to replicate the results on ImageNet multiple times over different random seeds as we did for CIFAR-10 and SVHN. Thus, we leave a more detailed investigation of catastrophic overfitting on ImageNet for future work.

2.11.4 Ablation studies

In this section, we aim to provide more details about sensitivity of `GradAlign` to its hyperparameter λ, the total number of training epochs, and also discuss training with `GradAlign` for very high ε values.

Ablation study for GradAlign λ. We provide an ablation study for the regularization parameter λ of `GradAlign` in Fig. 2.15, where we plot the adversarial accuracy of ResNet-18 trained using FGSM + `GradAlign` with $\varepsilon = 16/255$ on CIFAR-10. First, we observe that for small λ catastrophic overfitting occurs so that the average PGD-50-10 accuracy is either 0% or greater than 0% but has a high standard deviation since only some runs are successful while other runs fail because of catastrophic overfitting. We observe that the best performance is achieved for $\lambda = 2$ where catastrophic overfitting does not occur and the final adversarial accuracy is very concentrated. For larger λ values we observe a slow decrease in the adversarial accuracy since the model becomes overregularized. We note that the range of λ values which have close to the best performance ($\geq 26\%$ adversarial accuracy) ranges in $[0.25, 4]$, thus we conclude that `GradAlign` is robust to the exact choice of λ. This is also confirmed by our hyperparameter selection method for Fig. 2.8, where we performed a linear interpolation on the logarithmic scale between successful λ values for $\varepsilon = 8/255$ and $\varepsilon = 16/255$. Even such a coarse hyperparameter selection method, could ensure that none of the FGSM + `GradAlign` runs reported in Fig. 2.15 suffered from catastrophic overfitting.

Ablation study for the total number of training epochs. Recently, Rice et al. (2020) brought up the importance of early stopping in adversarial training. They identify the phenomenon called *robust overfitting* when training longer hurts the adversarial accuracy on the test set. Thus, we check here whether training with `GradAlign` has some influence on robust overfitting. We note that the authors of Rice et al. (2020) suggest that robust and catastrophic overfitting phenomena are distinct since robust overfitting implies a gap between training and test set robustness, while catastrophic overfitting implies

low robustness on *both* training and test sets. To explore this for FGSM + `GradAlign`, in Fig. 2.16 we show the final clean and adversarial accuracies for five different models trained with $\{30, 50, 100, 150, 250\}$ epochs. We observe the same trend as Rice et al. (2020) report: training longer slightly degrades adversarial accuracy (while in our case also the clean accuracy slightly improves). Thus, this experiment also suggests that robust overfitting is not directly connected to catastrophic overfitting and has to be addressed separately. Finally, we note based on Fig. 2.16 that when we use FGSM in combination with `GradAlign`, even training *up to 200 epochs* does not lead to catastrophic overfitting.

Ablation study for very high ε. Here we make an additional test on whether `GradAlign` prevents catastrophic overfitting for very high ε values. In Fig. 2.8 and Fig. 2.14 we showed results for $\varepsilon \leq 16$ for CIFAR-10 and for $\varepsilon \leq 12$ on SVHN. For SVHN, FGSM + `GradAlign` achieves 24.04±0.31% adversarial accuracy which is already close to that of a majority classifier (18.50%). The effect of increasing the perturbations size ε on SVHN even further just leads to learning a constant classifier. However, on CIFAR-10 for $\varepsilon = 16$, FGSM + `GradAlign` achieves 28.88±0.70% adversarial accuracy which is sufficiently far from that of a majority classifier (10.00%). Thus, a natural question is whether catastrophic overfitting still occurs for `GradAlign` on CIFAR-10, but just for higher ε values than what we considered in the main part of the paper. To show that it is not the case, in Table 2.7 we show the results of FGSM + `GradAlign` trained with $\varepsilon \in \{24/255, 32/255\}$ (we use $\lambda = 2.0$ and the maximum learning rate 0.1). We observe no signs of catastrophic overfitting *even for very high* ε such as $32/255$. Note that in this case the standard accuracy is very low (23.07±3.35%), thus considering such large perturbations is not practically interesting, but it rather serves as a sanity check that our method does not suffer from catastrophic overfitting even for very high ε.

Table 2.7: Robustness and accuracy of FGSM + `GradAlign` for very high ε on CIFAR-10 without early stopping for ResNet-18. We report results with the standard deviation and averaged over 3 random seeds used for training. We observe no catastrophic overfitting even for very high ε.

ℓ_∞-radius	Standard accuracy	PGD-50-10 accuracy
24/255	41.80±0.36%	17.07±0.90%
32/255	23.07±3.35%	12.93±1.44%

2.11.5 Comparison of GradAlign to gradient-based penalties

In this section, we compare `GradAlign` to other alternatives: ℓ_2 gradient norm penalization and CURE (Moosavi-Dezfooli et al., 2019b). The motivation to study them comes from the fact that after catastrophic overfitting, the input gradients change dramatically inside the ℓ_∞-balls around input points, and thus other gradient-based regularizers may also be able to improve the stability of the input gradients and thus prevent catastrophic overfitting.

In Table 2.8, we present results of FGSM training with other gradient-based penalties studied in the literature:

- ℓ_2 gradient norm regularization (Ross and Doshi-Velez, 2018; Simon-Gabriel et al., 2019): $\lambda \|\nabla_x \ell(x, y; \theta)\|_2^2$,
- curvature regularization (CURE) (Moosavi-Dezfooli et al., 2019b): $\lambda \|\nabla_x \ell(x + \delta_{FGSM}, y; \theta) - \nabla_x \ell(x, y; \theta)\|_2^2$.

First of all, we note that the originally proposed approaches (Ross and Doshi-Velez, 2018; Simon-Gabriel et al., 2019; Moosavi-Dezfooli et al., 2019b) *do not* involve adversarial training and rely *only* on these gradient penalties to achieve some degree of robustness. In contrast, we *combine* the gradient penalties with FGSM training to see whether they can prevent catastrophic overfitting similarly to `GradAlign`. For the gradient norm penalty, we use the regularization parameters $\lambda \in \{1{,}000, 2{,}000\}$ for $\varepsilon \in \{8/255, 16/255\}$ respectively. For CURE, we use $\lambda \in \{700, 20{,}000\}$ for $\varepsilon \in \{8/255, 16/255\}$ respectively. In both cases, we found the optimal hyperparameters using a grid search over λ. We can see that for $\varepsilon = 8/255$ all three approaches successfully prevent catastrophic overfitting, although the final robustness slightly varies between 46.69% for FGSM with the ℓ_2-gradient penalty and 47.58% for FGSM with `GradAlign`.

For $\varepsilon = 16/255$, both FGSM + CURE and FGSM + `GradAlign` prevent catastrophic overfitting leading to very concentrated results with a small standard deviation (0.29% and 0.70% respectively). However, the average adversarial accuracy is better for FGSM + `GradAlign`: 28.88% vs 25.38%. At the same time, FGSM with the ℓ_2-gradient penalty leads to unstable final performance: the adversarial accuracy has a high standard deviation: $13.64 \pm 11.2\%$.

We think that the main difference in the performance of `GradAlign` compared to the gradient penalties that we considered comes from the fact that it is invariant to the gradient norm, and it takes into account only the directions of two gradients inside the ℓ_∞-ball around the given input.

Inspired by CURE, we also tried two additional experiments:

1. Using the FGSM point δ_{FGSM} for the gradient taken at the second input point for `GradAlign`, but we observed that it does not make a substantial difference, i.e. this version of `GradAlign` also prevents catastrophic overfitting and leads to similar results. However, if we use CURE without FGSM in the cross-entropy loss, then we observe a benefit of using δ_{FGSM} in the regularizer which is consistent with the observations made in Moosavi-Dezfooli et al. (2019b).
2. Using `GradAlign` without FGSM in the cross-entropy loss. In this case, we observed that the model did not significantly improve its robustness suggesting that `GradAlign` *is not* a sufficient regularizer on its own to promote robustness and has to be used *with* some adversarial training method.

We think that an interesting future direction is to explore how one can speed up `GradAlign` or to come up with other regularization methods that are also able to prevent catastrophic

Table 2.8: Additional comparison of FGSM AT with `GradAlign` to FGSM AT with other gradient penalties on CIFAR-10. We report results without early stopping for ResNet-18. All the results are reported with the standard deviation and averaged over 5 random seeds used for training.

Model	**Accuracy**	
	Standard	PGD-50-10
	$\varepsilon = 8/255$	
FGSM + $\|\nabla_x\|_2^2$	77.47±0.14%	46.69±1.27%
FGSM + CURE	80.20±0.29%	47.25±0.21%
FGSM + `GradAlign`	81.00±0.37%	**47.58±0.24%**
	$\varepsilon = 16/255$	
FGSM + $\|\nabla_x\|_2^2$	56.44±2.22%	13.64±11.2%
FGSM + CURE	62.39±0.42%	25.38±0.29%
FGSM + `GradAlign`	58.46±0.22%	**28.88±0.70%**

overfitting, but avoid relying on the input gradients which lead to a slowdown in training. We think that some potential strategies to speed up `GradAlign` can include parallelization of the computations or saving some computations by subsampling the training batches for the regularizer. We postpone a further exploration of these ideas to future work.

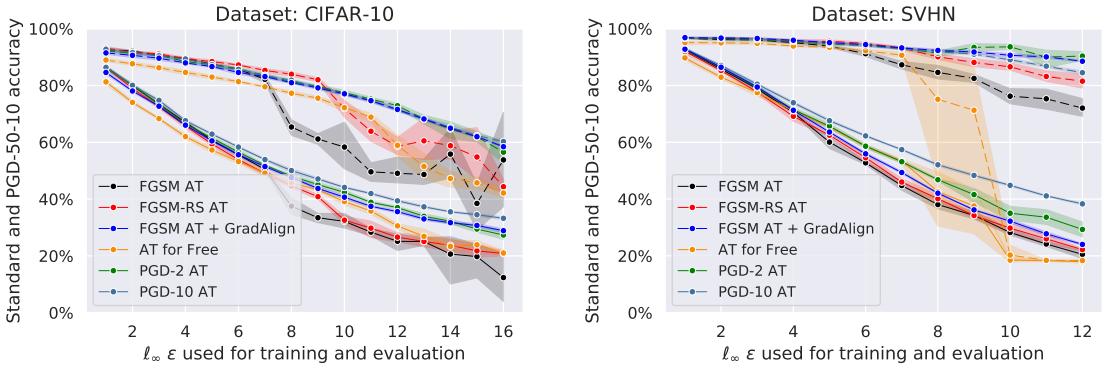


Figure 2.14: Accuracy (dashed line) and robustness (solid line) of different adversarial training (AT) methods on CIFAR-10 and SVHN with ResNet-18 trained and evaluated with different l_∞-radii. The results are obtained **with early stopping**, averaged over 5 random seeds used for training and reported with the standard deviation.

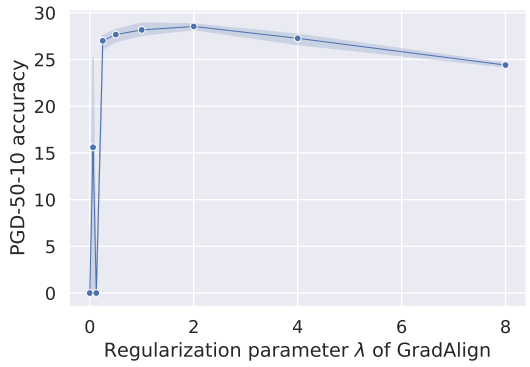


Figure 2.15: Ablation study for the regularization parameter λ for FGSM + `GradAlign` under $\varepsilon = {}^{16}/_{255}$ without early stopping. We train ResNet-18 models on CIFAR-10. The results are averaged over 3 random seeds used for training and reported with the standard deviation.

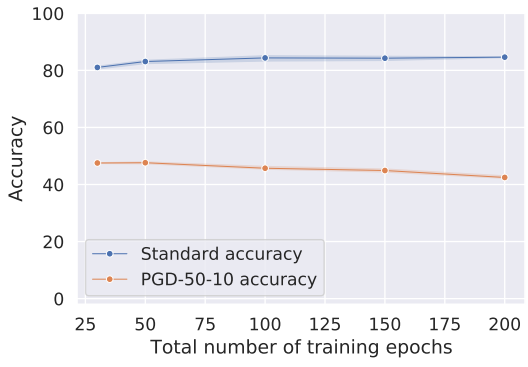


Figure 2.16: Ablation study for the total number of training epochs for FGSM + `GradAlign` under $\varepsilon = {}^{8}/_{255}$ without early stopping. We train ResNet-18 models on CIFAR-10. The results are averaged over 3 random seeds used for training and reported with the standard deviation.

3 Square Attack: a Query-Efficient Black-Box Adversarial Attack via Random Search

3.1 Preface

In this chapter, based on Andriushchenko et al. (2020), we focus on the problem of adversarial attacks on image classification models in the black-box setting, where the attacker has no access to the model's parameters or gradients.

Summary We propose the *Square Attack*, a score-based black-box l_2- and l_∞-adversarial attack that does not rely on local gradient information and thus is not affected by gradient masking. Square Attack is based on a randomized search scheme which selects localized square-shaped updates at random positions so that at each iteration the perturbation is situated approximately at the boundary of the feasible set. Our method is significantly more query efficient and achieves a higher success rate compared to the state-of-the-art methods, especially in the untargeted setting. In particular, on ImageNet we improve the average query efficiency in the untargeted setting for various deep networks by a factor of at least 1.8 and up to 3 compared to the recent state-of-the-art l_∞-attack of Al-Dujaili & O'Reilly (2020). Moreover, although our attack is *black-box*, it can also outperform gradient-based *white-box* attacks on the standard benchmarks achieving a new state-of-the-art in terms of the success rate. The code of our method is available at https://github.com/max-andr/square-attack.

Co-authors Francesco Croce, Nicolas Flammarion, Matthias Hein.

Contributions Maksym Andriushchenko proposed the project idea, designed and evaluated the l_∞ attack algorithm. Francesco Croce designed and evaluated the l_2 attack algorithm.

3.2 Introduction

Adversarial examples are of particular concern when it comes to applications of machine learning which are safety-critical. Many defenses against adversarial examples have been

proposed (Gu and Rigazio, 2015a; Zheng et al., 2016; Papernot et al., 2016b; Bastani et al., 2016; Madry et al., 2018; Akhtar and Mian, 2018; Biggio and Roli, 2018a) but with limited success, as new more powerful attacks could break many of them (Carlini and Wagner, 2017; Athalye et al., 2018; Mosbach et al., 2018; Chen et al., 2018; Zheng et al., 2019). In particular, gradient obfuscation or masking (Athalye et al., 2018; Mosbach et al., 2018) is often the reason why seemingly robust models turn out to be non-robust in the end. Gradient-based attacks are most often affected by this phenomenon (white-box attacks but also black-box attacks based on finite difference approximations (Mosbach et al., 2018)). Thus it is important to have attacks which are based on different principles. Black-box attacks have recently become more popular (Narodytska and Kasiviswanathan, 2017; Brendel et al., 2018a; Su et al., 2019) as their attack strategies are quite different from the ones employed for adversarial training, where often PGD-type attacks (Madry et al., 2018) are used. However, a big weakness currently is that these black-box attacks need to query the classifier too many times before they find adversarial examples, and their success rate is sometimes significantly lower than that of white-box attacks.

In this paper we propose Square Attack, a score-based adversarial attack, i.e. it can query the probability distribution over the classes predicted by a classifier but has no access to the underlying model. The Square Attack exploits random search[1] (Rastrigin, 1963; Schumer and Steiglitz, 1968) which is one of the simplest approaches for black-box optimization. Due to a particular sampling distribution, it requires significantly fewer queries compared to the state-of-the-art black-box methods (see Fig. 3.1) in the score-based threat model while outperforming them in terms of *success rate*, i.e. the percentage of successful adversarial examples. This is achieved by a combination of a particular initialization strategy and our square-shaped updates. We motivate why these updates are particularly suited to attack neural networks and provide convergence guarantees for a variant of our method. In an extensive evaluation with untargeted and targeted attacks, three datasets (MNIST, CIFAR-10, ImageNet), normal and robust models, we show that Square Attack outperforms state-of-the-art methods in the l_2- and l_∞-threat model.

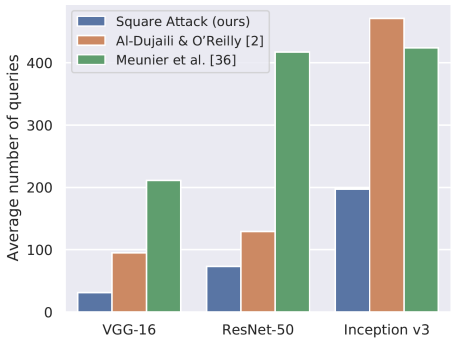


Figure 3.1: Avg. number of queries of successful untargeted l_∞-attacks on three ImageNet models for three score-based black-box attacks. Square Attack outperforms all other attacks by large margin

3.3 Related Work

We discuss black-box attacks with l_2- and l_∞-perturbations since our attack focuses on this setting. Although attacks for other norms, e.g. l_0, exist (Narodytska and Kasiviswanathan, 2017; Croce and Hein, 2019), they are often algorithmically different due

[1]It is an iterative procedure different from random sampling inside the feasible region.

to the geometry of the perturbations.

l_2- and l_∞-score-based attacks. Score-based black-box attacks have only access to the score predicted by a classifier for each class for a given input. Most of such attacks in the literature are based on gradient estimation through finite differences. The first papers in this direction Bhagoji et al. (2018); Ilyas et al. (2018); Uesato et al. (2018) propose attacks which approximate the gradient by sampling from some noise distribution around the point. While this approach can be successful, it requires many queries of the classifier, particularly in high-dimensional input spaces as in image classification. Thus, improved techniques reduce the dimension of the search space via using the principal components of the data (Bhagoji et al., 2018), searching for perturbations in the latent space of an auto-encoder (Tu et al., 2019) or using a low-dimensional noise distribution Ilyas et al. (2019a). Other attacks exploit evolutionary strategies or random search, e.g., Alzantot et al. (2019) use a genetic algorithm to generate adversarial examples and alleviate gradient masking as they can reduce the robust accuracy on randomization- and discretization-based defenses. The l_2-attack of Guo et al. (2019b) can be seen as a variant of random search which chooses the search directions in an orthonormal basis and tests up to two candidate updates at each step. However, their algorithm can have suboptimal query efficiency since it adds at every step only small (in l_2-norm) modifications, and suboptimal updates cannot be undone as they are orthogonal to each other. A recent line of work has pursued black-box attacks which are based on the observation that successful adversarial perturbations are attained at corners of the l_∞-ball intersected with the image space $[0, 1]^d$ (Seungyong et al., 2019; Al-Dujaili and O'Reilly, 2020; Meunier et al., 2019). Searching over the corners allows to apply discrete optimization techniques to generate adversarial attacks, significantly improving the query efficiency. Both Seungyong et al. (2019) and Al-Dujaili and O'Reilly (2020) divide the image according to some coarse grid, perform local search in this lower dimensional space allowing componentwise changes only of $-\epsilon$ and ϵ, then refine the grid and repeat the scheme. In Al-Dujaili and O'Reilly (2020) such a procedure is motivated as an estimation of the gradient signs. Recently, Meunier et al. (2019) proposed several attacks based on evolutionary algorithms, using discrete and continuous optimization, achieving nearly state-of-the-art query efficiency for the l_∞-norm. In order to reduce the dimensionality of the search space, they use the "tiling trick" of Ilyas et al. (2019a) where they divide the perturbation into a set of squares and modify the values in these squares with evolutionary algorithms. A related idea also appeared earlier in Fawzi and Frossard (2016) where they introduced black rectangle-shaped perturbations for generating adversarial occlusions. In Meunier et al. (2019), as in Ilyas et al. (2019a), both size and position of the squares are fixed at the beginning and not optimized. Despite their effectiveness for the l_∞-norm, these discrete optimization based attacks are not straightforward to adapt to the l_2-norm. Finally, approaches based on Bayesian optimization exist, e.g., Shukla et al. (2019), but show competitive performance only in a low-query regime.

Different threat and knowledge models. We focus on l_p-*norm-bounded* adversarial perturbations (for other perturbations such as rotations, translations, occlusions in the

black-box setting see, e.g., Fawzi and Frossard (2016)). Perturbations with *minimal* l_p-norm are considered in Chen et al. (2017); Tu et al. (2019) but require significantly more queries than norm-bounded ones. Thus we do not compare to them, except for Guo et al. (2019b) which has competitive query efficiency while aiming at small perturbations.

In other cases the attacker has a different knowledge of the classifier. A more restrictive scenario, considered by *decision-based* attacks (Brendel et al., 2018a; Cheng et al., 2019a; Guo et al., 2019a; Brunner et al., 2019; Chen et al., 2019), is when the attacker can query only the decision of the classifier, but not the predicted scores. Other works use more permissive threat models, e.g., when the attacker already has a substitute model similar to the target one (Papernot et al., 2016a; Yan et al., 2019; Cheng et al., 2019b; Du et al., 2020; Suya et al., 2019) and thus can generate adversarial examples for the substitute model and then transfer them to the target model. Related to this, Yan et al. (2019) suggest to refine this approach by running a black-box gradient estimation attack in a subspace spanned by the gradients of substitute models. However, the gain in query efficiency given by such extra knowledge does not account for the computational cost required to train the substitute models, particularly high on ImageNet-scale. Finally, Li et al. (2019b) use extra information on the target data distribution to train a model that predicts adversarial images that are then refined by gradient estimation attacks.

3.4 Square Attack

In the following we recall the definitions of the adversarial examples in the threat model we consider and present our black-box attacks for the l_∞- and l_2-norms.

3.4.1 Adversarial Examples in the l_p-threat Model

Let $f : [0,1]^d \to \mathbb{R}^K$ be a classifier, where d is the input dimension, K the number of classes and $f_k(x)$ is the predicted score that x belongs to class k. The classifier assigns class $\arg\max_{k=1,\dots,K} f_k(x)$ to the input x. The goal of an *untargeted* attack is to change the correctly predicted class y for the point x. A point $\hat{x}$ is called an *adversarial example* with an l_p-norm bound of ϵ for x if

$$\arg\max_{k=1,\dots,K} f_k(\hat{x}) \neq y, \quad \|\hat{x} - x\|_p \leq \epsilon \quad \text{and} \quad \hat{x} \in [0,1]^d,$$

where we have added the additional constraint that $\hat{x}$ is an image. The task of finding $\hat{x}$ can be rephrased as solving the constrained optimization problem

$$\min_{\hat{x} \in [0,1]^d} L(f(\hat{x}), y), \quad \text{s.t.} \quad \|\hat{x} - x\|_p \leq \epsilon, \tag{3.1}$$

for a loss L. In our experiments, we use $L(f(\hat{x}), y) = f_y(\hat{x}) - \max_{k \neq y} f_k(\hat{x})$.

The goal of *targeted* attacks is instead to change the decision of the classifier to a particular

Algorithm 1: The Square Attack via random search

Input: classifier f, point $x \in \mathbb{R}^d$, image size w, number of color channels c, l_p-radius ϵ, label $y \in \{1, \dots, K\}$, number of iterations N

Output: approximate minimizer $\hat{x} \in \mathbb{R}^d$ of the problem stated in Eq. (3.1)

$\hat{x} \leftarrow init(x)$, $l^* \leftarrow L(f(x), y)$, $i \leftarrow 1$

while $i < N$ ***and*** $\hat{x}$ *is not adversarial* **do**

  $h^{(i)} \leftarrow$ side length of the square to modify (according to some schedule)

  $\delta \sim P(\epsilon, h^{(i)}, w, c, \hat{x}, x)$ (see Alg. 2 and 3 for the sampling distributions)

  $\hat{x}_{\text{new}} \leftarrow$ Project $\hat{x} + \delta$ onto $\{z \in \mathbb{R}^d : \|z - x\|_p \leq \epsilon\} \cap [0, 1]^d$

  $l_{\text{new}} \leftarrow L(f(\hat{x}_{\text{new}}), y)$

  if $l_{new} < l^*$ **then** $\hat{x} \leftarrow \hat{x}_{\text{new}}$, $l^* \leftarrow l_{\text{new}}$;

  $i \leftarrow i + 1$

end

class t, i.e., to find $\hat{x}$ so that $\arg\max_k f_k(\hat{x}) = t$ under the same constraints on $\hat{x}$. We further discuss the targeted attacks in Sup. 3.12.1.

3.4.2 General Algorithmic Scheme of the Square Attack

Square Attack is based on *random search* which is a well known iterative technique in optimization introduced by Rastrigin in 1963 (Rastrigin, 1963). The main idea of the algorithm is to sample a random update δ at each iteration, and to add this update to the current iterate $\hat{x}$ if it improves the objective function. Despite its simplicity, random search performs well in many situations (Zabinsky, 2010) and does not depend on gradient information from the objective function g.

Many variants of random search have been introduced (Matyas, 1965; Schumer and Steiglitz, 1968; Schrack and Choit, 1976), which differ mainly in how the random perturbation is chosen at each iteration (the original scheme samples uniformly on a hypersphere of fixed radius). For our goal of crafting adversarial examples we come up with two sampling distributions specific to the l_∞- and the l_2-attack (Sec. 3.4.3 and Sec. 3.4.4), which we integrate in the classic random search procedure. These sampling distributions are motivated by both how images are processed by neural networks with convolutional filters and the shape of the l_p-balls for different p. Additionally, since the considered objective is non-convex when using neural networks, a good initialization is particularly important. We then introduce a specific one for better query efficiency.

Our proposed scheme differs from classical random search by the fact that the perturbations $\hat{x} - x$ are constructed such that for every iteration they lie on the boundary of the l_∞- or l_2-ball before projection onto the image domain $[0, 1]^d$. Thus we are using the perturbation budget almost maximally at each step. Moreover, the changes are localized in the image in the sense that at each step we modify just a small fraction of contiguous pixels shaped into **squares**. Our overall scheme is presented in Algorithm 1. First, the

algorithm picks the side length $h^{(i)}$ of the square to be modified (step 3), which is decreasing according to an a priori fixed schedule. This is in analogy to the step-size reduction in gradient-based optimization. Then in step 4 we sample a new update δ and add it to the current iterate (step 5). If the resulting loss (obtained in step 6) is smaller than the best loss so far, the change is accepted otherwise discarded. Since we are interested in query efficiency, the algorithm stops as soon as an adversarial example is found. The time complexity of the algorithm is dominated by the evaluation of $f(\hat{x}_{\text{new}})$, which is performed at most N times, with N total number of iterations. We plot the resulting adversarial perturbations in Fig. 3.3 and additionally in Sup. 3.12 where we also show imperceptible perturbations.

We note that previous works (Ilyas et al., 2019a; Seungyong et al., 2019; Meunier et al., 2019) generate perturbations containing squares. However, while those use a fixed grid on which the squares are constrained, we optimize the position of the squares as well as the color, making our attack more flexible and effective. Moreover, unlike previous works, we motivate squared perturbations with the structure of the convolutional filters (see Sec. 3.5).

Size of the squares. Given images of size $w \times w$, let $p \in [0, 1]$ be the percentage of elements of x to be modified. The length h of the side of the squares used is given by the closest positive integer to $\sqrt{p \cdot w^2}$ (and $h \geq 3$ for the l_2-attack). Then, the initial p is the only free parameter of our scheme. With $N = 10000$ iterations available, we halve the value of p at $i \in \{10, 50, 200, 1000, 2000, 4000, 6000, 8000\}$ iterations. For different N we rescale the schedule accordingly.

3.4.3 The l_∞-Square Attack

Initialization. As initialization we use vertical stripes of width one where the color of each stripe is sampled uniformly at random from $\{-\epsilon, \epsilon\}^c$ (c number of color channels). We found that convolutional networks are particularly sensitive to such perturbations, see also Yin et al. (2019) for a detailed discussion on the sensitivity of neural networks to various types of high frequency perturbations.

Sampling distribution. Similar to Seungyong et al. (2019) we observe that successful l_∞-perturbations usually have values $\pm\epsilon$ in all the components (note that this does not hold perfectly due to the image constraints $\hat{x} \in [0, 1]^d$). In particular, it holds

$$\hat{x}_i \in \{\max\{0, x_i - \epsilon\}, \min\{1, x_i + \epsilon\}\}.$$

Algorithm 2: Sampling distribution P for l_∞-norm

Input: maximal norm ϵ, window size h, image size w, color channels c
Output: New update δ

$\delta \leftarrow$ array of zeros of size $w \times w \times c$
sample uniformly $r, s \in \{0, \dots, w-h\} \subset \mathbb{N}$
for $i = 1, \dots, c$ **do**
  $\rho \leftarrow Uniform(\{-2\epsilon, 2\epsilon\})$
  $\delta_{r+1:r+h,\, s+1:s+h,\, i} \leftarrow \rho \cdot \mathbb{1}_{h \times h}$
end

Our sampling distribution P for the l_∞-norm described in Algorithm 2 selects sparse updates of $\hat{x}$ with $\|\delta\|_0 = h \cdot h \cdot c$ where $\delta \in \{-2\epsilon, 0, 2\epsilon\}^d$ and the non-zero elements are grouped to form a square. In this way, after the projection onto the l_∞-ball of radius ϵ (Step 5 of Algorithm 1) all components i for which $\epsilon \leq x_i \leq 1 - \epsilon$ satisfy $\hat{x}_i \in \{x_i - \epsilon, x_i + \epsilon\}$, i.e. differ from the original point x in each element either by ϵ or $-\epsilon$. Thus $\hat{x} - x$ is situated at one of the corners of the l_∞-ball (modulo the components which are close to the boundary). Note that all projections are done by clipping. Moreover, we fix the elements of δ belonging to the same color channel to have the same sign, since we observed that neural networks are particularly sensitive to such perturbations (see Sec. 3.5.3).

3.4.4 The l_2-Square Attack

Initialization. The l_2-perturbation is initialized by generating a 5×5 grid-like tiling by squares of the image, where the perturbation on each tile has the shape described next in the sampling distribution. The resulting perturbation $\hat{x} - x$ is rescaled to have l_2-norm ϵ and the resulting $\hat{x}$ is projected onto $[0, 1]^d$ by clipping.

Sampling distribution. First, let us notice that the adversarial perturbations typically found for the l_2-norm tend to be much more localized than those for the l_∞-norm (Tsipras et al., 2019), in the sense that large changes are applied on some pixels of the original image, while many others are minimally modified. To mimic this feature we introduce a new update η which has two "centers" with large absolute value and opposite signs, while the other components have lower absolute values as one gets farther away from the centers, but never reaching zero (see Fig. 3.2 for one example with $h = 8$ of the resulting update η). In this way the modifications are localized and with high contrast between the different halves, which we found to improve the query efficiency. Concretely, we define $\eta^{(h_1,h_2)} \in \mathbb{R}^{h_1 \times h_2}$

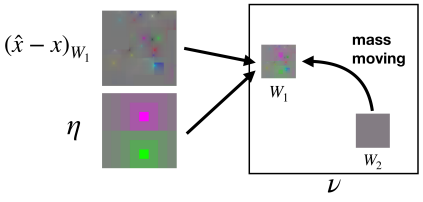


Figure 3.2: Perturbation of the l_2-attack

Algorithm 3: Sampling distribution P for l_2-norm

Input: maximal norm ϵ, window size h, image size w, number of color channels c, current image $\hat{x}$, original image x

Output: New update δ

$\nu \leftarrow \hat{x} - x$

sample uniformly $r_1, s_1, r_2, s_2 \in \{0, \dots, w-h\}$

$W_1 := r_1 + 1 : r_1 + h, s_1 + 1 : s_1 + h$, $W_2 := r_2 + 1 : r_2 + h, s_2 + 1 : s_2 + h$

$\epsilon^2_{\text{unused}} \leftarrow \epsilon^2 - \|\nu\|_2^2$, $\quad \eta^* \leftarrow \eta / \|\eta\|_2$ with η as in (3.2)

for $i = 1, \dots, c$ **do**

  $\rho \leftarrow Uniform(\{-1, 1\})$

  $\nu_{\text{temp}} \leftarrow \rho\eta^* + \nu_{W_1,i} / \|\nu_{W_1,i}\|_2$

  $\epsilon^i_{\text{avail}} \leftarrow \sqrt{\|\nu_{W_1 \cup W_2,i}\|_2^2 + \epsilon^2_{\text{unused}}/c}$

  $\nu_{W_2,i} \leftarrow 0$, $\quad \nu_{W_1,i} \leftarrow (\nu_{\text{temp}} / \|\nu_{\text{temp}}\|_2)\epsilon^i_{\text{avail}}$

end

$\delta \leftarrow x + \nu - \hat{x}$

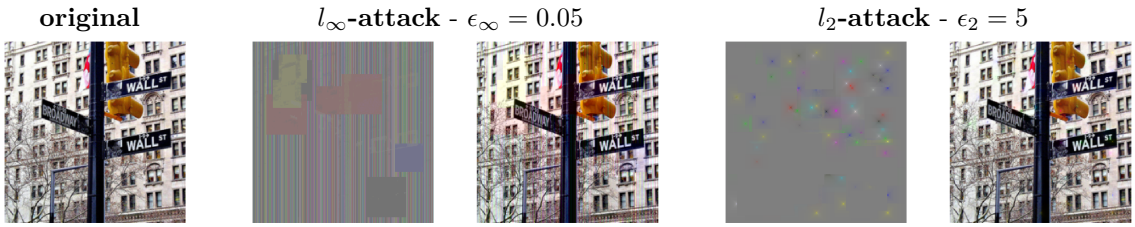


Figure 3.3: Visualization of the adversarial perturbations and examples found by the l_∞- and l_2-versions of the Square Attack on ResNet-50

(for some $h_1, h_2 \in \mathbb{N}_+$ such that $h_1 \geq h_2$) for every $1 \leq r \leq h_1$, $1 \leq s \leq h_2$ as

$$\eta^{(h_1,h_2)}_{r,s} = \sum_{k=0}^{M(r,s)} \frac{1}{(n+1-k)^2}, \quad \text{with} \quad n = \left\lfloor \frac{h_1}{2} \right\rfloor,$$

and $M(r,s) = n - \max\{|r - \left\lfloor \frac{h_1}{2} \right\rfloor - 1|, |s - \left\lfloor \frac{h_2}{2} \right\rfloor - 1|\}$. The intermediate square update $\eta \in \mathbb{R}^{h \times h}$ is then selected uniformly at random from either

$$\eta = \left(\eta^{(h,k)}, -\eta^{(h,h-k)}\right), \quad \text{with } k = \lfloor h/2 \rfloor, \tag{3.2}$$

or its transpose (corresponding to a rotation of 90°).

Second, unlike l_∞-constraints, l_2-constraints do not allow to perturb each component independently from the others as the overall l_2-norm must be kept smaller than ϵ. Therefore, to modify a perturbation $\hat{x} - x$ of norm ϵ with localized changes while staying on the hypersphere, we have to "move the mass" of $\hat{x} - x$ from one location to another. Thus, our scheme consists in randomly selecting two squared windows in the current perturbation $\nu = \hat{x} - x$, namely ν_{W_1} and ν_{W_2}, setting $\nu_{W_2} = 0$ and using the budget of $\|\nu_{W_2}\|_2$ to increase the total perturbation of ν_{W_1}. Note that the perturbation of W_1 is then a com-

bination of the existing perturbation plus the new generated η. We report the details of this scheme in Algorithm 3 where step 4 allows to utilize the budget of l_2-norm lost after the projection onto $[0,1]^d$. The update δ output by the algorithm is such that the next iterate $\hat{x}_{\text{new}} = \hat{x} + \delta$ (before projection onto $[0,1]^d$ by clipping) belongs to the hypersphere $B_2(x, \epsilon)$ as stated in the following proposition.

Proposition 3.4.1. *Let δ be the output of Algorithm 3. Then $\|\hat{x} + \delta - x\|_2 = \epsilon$.*

3.5 Theoretical and Empirical Justification of the Method

We provide high-level theoretical justifications and empirical evidence regarding the algorithmic choices in Square Attack, with focus on the l_∞-version (the l_2-version is significantly harder to analyze).

3.5.1 Convergence Analysis of Random Search

First, we want to study the convergence of the random search algorithm when considering an L-smooth objective function g (such as neural networks with activation functions like softplus, swish, ELU, etc) on the whole space $\mathbb{R}^d$ (without projection[2]) under the following assumptions on the update δ_t drawn from the sampling distribution P_t:

$$\mathbb{E}\|\delta_t\|_2^2 \leq \gamma_t^2 C \text{ and } \mathbb{E}|\langle \delta_t, v\rangle| \geq \tilde{C}\gamma_t \|v\|_2, \ \forall v \in \mathbb{R}^d, \tag{3.3}$$

where γ_t is the step size at iteration t, $C, \tilde{C} > 0$ some constants and $\langle \cdot, \cdot \rangle$ denotes the inner product. We obtain the following result, similar to existing convergence rates for zeroth-order methods (Nemirovsky and Yudin, 1983; Nesterov and Spokoiny, 2017; Duchi et al., 2015):

Proposition 3.5.1. *Suppose that $\mathbb{E}[\delta_t] = 0$ and the assumptions in Eq. (3.3) hold. Then for step-sizes $\gamma_t = \gamma/\sqrt{T}$, we have*

$$\min_{t=0,\dots,T} \mathbb{E}\|\nabla g(x_t)\|_2 \leq \frac{2}{\gamma \tilde{C}\sqrt{T}} \left(g(x_0) - \mathbb{E}g(x_{T+1}) + \frac{\gamma^2 C L}{2} \right).$$

This basically shows for T large enough one can make the gradient arbitrary small, meaning that the random search algorithm converges to a critical point of g (one cannot hope for much stronger results in non-convex optimization without stronger conditions).

Unfortunately, the second assumption in Eq. (3.3) does not directly hold for our sampling distribution P for the l_∞-norm (see Sup. 3.8.3), but holds for a similar one, P^{multiple}, where each component of the update δ is drawn uniformly at random from $\{-2\epsilon, 2\epsilon\}$. In

[2]Nonconvex constrained optimization under noisy oracles is notoriously harder (Davis and Drusvyatskiy, 2019).

fact we show in Sup. 3.8.4, using the Khintchine inequality (Haagerup, 1981), that

$$\mathbb{E}\|\delta_t\|_2^2 \leq 4c\varepsilon^2 h^2 \text{ and } \mathbb{E}|\langle \delta_t, v\rangle| \geq \frac{\sqrt{2}c\varepsilon h^2}{d}\|v\|_2, \ \forall v \in \mathbb{R}^d.$$

Moreover, while P^{multiple} performs worse than the distribution used in Algorithm 2, we show in Sec. 3.5.3 that it already reaches state-of-the-art results.

3.5.2 Why Squares?

Previous works (Seungyong et al., 2019; Meunier et al., 2019) build their l_∞-attacks by iteratively adding square modifications. Likewise we change square-shaped regions of the image for both our l_∞- and l_2-attacks—with the difference that we can sample any square subset of the input, while the grid of the possible squares is fixed in Seungyong et al. (2019); Meunier et al. (2019). This leads naturally to wonder why squares are superior to other shapes, e.g. rectangles.

Let us consider the l_∞-threat model, with bound ϵ, input space $\mathbb{R}^{d\times d}$ and a convolutional filter $w \in \mathbb{R}^{s\times s}$ with entries unknown to the attacker. Let $\delta \in \mathbb{R}^{d\times d}$ be the sparse update with $\|\delta\|_0 = k \geq s^2$ and $\|\delta\|_\infty \leq \epsilon$. We denote by $S(a, b)$ the index set of the rectangular support of δ with $|S(a, b)| = k$ and shape $a \times b$. We want to provide intuition why sparse square-shaped updates are superior to rectangular ones in the sense of reaching a maximal change in the activations of the first convolutional layer.

Let $z = \delta * w \in \mathbb{R}^{d\times d}$ denote the output of the convolutional layer for the update δ. The l_∞-norm of z is the maximal componentwise change of the convolutional layer:

$$\begin{aligned}\|z\|_\infty &= \max_{u,v} |z_{u,v}| = \max_{u,v} \left| \sum_{i,j=1}^{s} \delta_{u-\lfloor \frac{s}{2}\rfloor+i, v-\lfloor \frac{s}{2}\rfloor+j} \cdot w_{i,j} \right| \\ &\leq \max_{u,v} \epsilon \sum_{i,j} |w_{i,j}| \mathbb{1}_{(u-\lfloor \frac{s}{2}\rfloor+i, v-\lfloor \frac{s}{2}\rfloor+j)\in S(a,b)},\end{aligned}$$

where elements with indices exceeding the size of the matrix are set to zero. Note that the indicator function attains 1 only for the non-zero elements of δ involved in the convolution to get $z_{u,v}$. Thus, to have the largest upper bound possible on $|z_{u,v}|$, for some (u, v), we need the largest possible amount of components of δ with indices in

$$C(u, v) = \left\{(u - \lfloor \frac{s}{2}\rfloor + i, v - \lfloor \frac{s}{2}\rfloor + j) : \ i, j = 1, \ldots, s\right\}$$

to be non-zero (that is in $S(a, b)$).

Therefore, it is desirable to have the shape $S(a, b)$ of the perturbation δ selected so to maximize the number N of convolutional filters $w \in \mathbb{R}^{s\times s}$ which fit into the rectangle $a \times b$. Let $\mathcal{F}$ be the family of the objects that can be defined as the union of axis-aligned rectangles with vertices on $\mathbb{N}^2$, and $\mathcal{G} \subset \mathcal{F}$ be the squares of $\mathcal{F}$ of shape $s \times s$ with $s \geq 2$.

Table 3.1: Ablation study of the l_∞-Square Attack which shows how the individual design decisions improve the performance. The fourth row corresponds to the method for which we have shown convergence guarantees in Sec. 3.5.1. The last row corresponds to our final l_∞-attack. c indicates the number of color channels, h the length of the side of the squares, so that "# random sign" c represents updates with constant sign for each color, while $c \cdot h^2$ updates with signs sampled independently of each other

Update shape	# random signs	Initialization	Failure rate	Avg. queries	Median queries
random	$c \cdot h^2$	vert. stripes	**0.0%**	401	48
random	$c \cdot h^2$	uniform rand.	**0.0%**	393	132
random	c	vert. stripes	**0.0%**	339	53
square	$c \cdot h^2$	vert. stripes	**0.0%**	153	15
rectangle	c	vert. stripes	**0.0%**	93	16
square	c	uniform rand.	**0.0%**	91	26
square	c	vert. stripes	**0.0%**	**73**	**11**

We have the following proposition:

Proposition 3.5.2. *Among the elements of $\mathcal{F}$ with area $k \geq s^2$, those which contain the largest number of elements of $\mathcal{G}$ have*

$$N^* = (a - s + 1)(b - s + 1) + (r - s + 1)^+ \tag{3.4}$$

of them, with $a = \left\lfloor \sqrt{k} \right\rfloor$, $b = \left\lfloor \frac{k}{a} \right\rfloor$, $r = k - ab$ and $z^+ = \max\{z, 0\}$.

This proposition states that, if we can modify only k elements of δ, then shaping them to form (approximately) a square allows to maximize the number of pairs (u, v) for which $|S(a, b) \cap C(u, v)| = s^2$. If $k = l^2$ then $a = b = l$ are the optimal values for the shape of the perturbation update, i.e. the shape is exactly a square.

3.5.3 Ablation Study

We perform an ablation study to show how the individual design decisions for the sampling distribution of the random search improve the performance of l_∞-Square Attack, confirming the theoretical arguments above. The comparison is done for an l_∞-threat model of radius $\epsilon = 0.05$ on $1,000$ test points for a ResNet-50 model trained normally on ImageNet (see Sec. 3.6 for details) with a query limit of $10,000$ and results are shown in Table 3.1. Our sampling distribution is special in two aspects: i) we use localized update shapes in form of squares and ii) the update is constant in each color channel. First, one can observe that our update shape "square" performs better than "rectangle" as we discussed in the previous section, and it is significantly better than "random" (the same amount of pixels is perturbed, but selected randomly in the image). This holds both for c (constant sign per color channel) and $c \cdot h^2$ (every pixel and color channel is changed independently of each other), with an improvement in terms of average queries of 339 to 73 and 401 to 153 respectively. Moreover, with updates of the same shape, the constant sign over color channels is better than selecting it uniformly at random (improvement in average queries:

Table 3.2: Results of **untargeted** attacks on ImageNet with a limit of 10,000 queries. For the l_∞-attack we set the norm bound $\epsilon = 0.05$ and for the l_2-attack $\epsilon = 5$. Models: normally trained **I**: Inception v3, **R**: ResNet-50, **V**: VGG-16-BN. The Square Attack outperforms for both threat models all other methods in terms of success rate and query efficiency. The missing entries correspond to the results taken from the original paper where some models were not reported

Norm	Attack	Failure rate			Avg. queries			Med. queries		
		I	R	V	I	R	V	I	R	V
l_∞	Bandits	3.4%	1.4%	2.0%	957	727	394	218	136	36
	Parsimonious	1.5%	-	-	722	-	-	237	-	-
	DFO_c–CMA	0.8%	**0.0%**	0.1%	630	270	219	259	143	107
	DFO_d–Diag. CMA	2.3%	1.2%	0.5%	424	417	211	**20**	20	2
	SignHunter	1.0%	0.1%	0.3%	471	129	95	95	39	43
	Square Attack	**0.3%**	**0.0%**	**0.0%**	**197**	**73**	**31**	24	**11**	**1**
l_2	Bandits	9.8%	6.8%	10.2%	1486	939	511	660	392	196
	SimBA-DCT	35.5%	12.7%	7.9%	**651**	**582**	452	564	467	360
	Square Attack	**7.1%**	**0.7%**	**0.8%**	1100	616	**377**	**385**	**170**	**109**

401 to 339 and 153 to 73). In total the algorithm with "square-c" needs more than 5× less average queries than "random-$c \cdot h^2$", showing that our sampling distribution is the key to the high query efficiency of Square Attack.

The last innovation of our random search scheme is the initialization, crucial element of every non-convex optimization algorithm. Our method ("square-c") with the vertical stripes initialization improves over a uniform initialization on average by $\approx 25\%$ and, especially, median number of queries (more than halved).

We want to also highlight that the sampling distribution "square-$c \cdot h^2$" for which we shown convergence guarantees in Sec. 3.5.1 performs already better in terms of the success rate and the median number of queries than the state of the art (see Sec. 3.6). For a more detailed ablation, also for our l_2-attack, see Sup. 3.10.

3.6 Experiments

In this section we show the effectiveness of the Square Attack. Here we concentrate on **untargeted** attacks since our primary goal is query efficient robustness evaluation, while the **targeted** attacks are postponed to the supplement. First, we follow the standard setup (Ilyas et al., 2019a; Meunier et al., 2019) of comparing black-box attacks on three ImageNet models in terms of success rate and query efficiency for the l_∞- and l_2-untargeted attacks (Sec. 3.6.1). Second, we show that our *black-box* attack can even outperform *white-box* PGD attacks on several models (Sec. 3.6.2). Finally, in the supplement we provide more experimental details (Sup. 3.9), a stability study of our attack for different parameters (Sup. 3.10) and random seeds (Sup. 3.11), and additional results including the experiments for targeted attacks (Sup. 3.12).

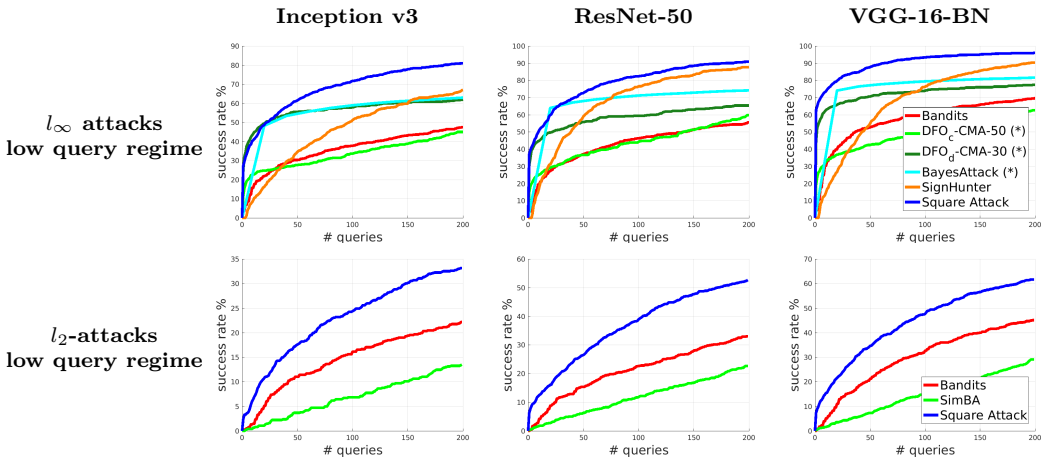


Figure 3.4: Success rate in the low-query regime (up to 200 queries). * denotes the results obtained via personal communication with the authors and evaluated on 500 and 10,000 randomly sampled points for BayesAttack (Shukla et al., 2019) and DFO (Meunier et al., 2019) methods, respectively

3.6.1 Evaluation on ImageNet

We compare the Square Attack to state-of-the-art score-based black-box attacks (without any extra information such as surrogate models) on three pretrained models in PyTorch (Inception v3, ResNet-50, VGG-16-BN) using 1,000 images from the ImageNet validation set. Unless mentioned otherwise, we use the code from the other papers with their suggested parameters. As it is standard in the literature, we give a budget of 10,000 queries per point to find an adversarial perturbation of l_p-norm at most ϵ. We report the *average* and *median* number of queries each attack requires to craft an adversarial example, together with the *failure rate*. All query statistics are computed only for successful attacks on the points which were originally correctly classified.

Tables 3.2 and 3.3 show that the Square Attack, despite its simplicity, achieves in all the cases (models and norms) the **lowest failure rate**, ($< 1\%$ everywhere except for the l_2-attack on Inception v3), and almost always requires **fewer queries** than the competitors to succeed. Fig. 3.4 shows the progression of the success rate of the attacks over the first 200 queries. Even in the low query regime the Square Attack outperforms the competitors for both norms. Finally, we highlight that the only hyperparameter of our attack, p, regulating the size of the squares, is set for all the models to 0.05 for l_∞ and 0.1 for l_2-perturbations.

l_∞-attacks. We compare our attack to Bandits (Ilyas et al., 2019b), Parsimonious (Seungyong et al., 2019), DFO$_c$ / DFO$_d$ (Meunier et al., 2019), and SignHunter (Al-Dujaili and O'Reilly, 2020). In Table 3.2 we report the results of the l_∞-attacks with norm bound of $\epsilon = 0.05$. The Square Attack always has the lowest failure rate, notably 0.0% in 2 out of 3 cases, and the lowest query consumption. Interestingly, our attack has median equal 1 on VGG-16-BN, meaning that the proposed initialization is particularly effective for this model.

The closest competitor in terms of the *average* number of queries is SignHunter (Al-Dujaili and O'Reilly, 2020), which still needs on average between 1.8 and 3 times more queries to find adversarial examples and has a higher failure rate than our attack. Moreover, the median number of queries of SignHunter is much worse than for our method (e.g. 43 vs 1 on VGG). We note that although DFO$_c$–CMA (Meunier et al., 2019) is competitive to our attack in terms of *median* queries, it has a significantly higher failure rate and between 2 and 7 times worse average number of queries. Additionally, our method is also more effective in the low-query regime (Fig. 3.4) than other methods (including Shukla et al. (2019)) on all the models.

Table 3.3: Query statistics for untargeted l_2-attacks computed for the points for which all three attacks are successful for fair comparison

Attack	Avg. queries			Med. queries		
	I	R	V	I	R	V
Bandits	536	635	398	368	314	177
SimBA-DCT	647	563	421	552	446	332
Square Attack	**352**	**287**	**217**	**181**	**116**	**80**

l_2-attacks. We compare our attack to Bandits (Ilyas et al., 2019a) and SimBA (Guo et al., 2019b) for $\epsilon = 5$, while we do not consider SignHunter (Al-Dujaili and O'Reilly, 2020) since it is not as competitive as for the l_∞-norm, and in particular worse than Bandits on ImageNet (see Fig. 2 in Al-Dujaili and O'Reilly (2020)).

As Table 3.2 and Fig. 3.4 show, the Square Attack outperforms by a large margin the other methods in terms of failure rate, and achieves the lowest median number of queries for all the models and the lowest average one for VGG-16-BN. However, since it has a significantly lower failure rate, the statistics of the Square Attack are biased by the "hard" cases where the competitors fail. Then, we recompute the same statistics considering only the points where all the attacks are successful (Table 3.3). In this case, our method improves by at least 1.5× the average and by at least 2× the median number of queries.

3.6.2 Square Attack Can be More Accurate than White-box Attacks

Here we test our attack on problems which are challenging for both white-box PGD and other black-box attacks. We use for evaluation *robust accuracy*, defined as the worst-case accuracy of a classifier when an attack perturbs each input in some l_p-ball. We show that our algorithm outperforms the competitors both on state-of-the-art robust models and defenses that induce different types of gradient masking. Thus, our attack is useful to evaluate robustness without introducing adaptive attacks designed for each model separately.

Table 3.4: On the robust models of Madry et al. (2018) and (Zhang et al., 2019b) on MNIST l_∞-Square Attack with $\epsilon = 0.3$ achieves state-of-the-art (SOTA) results outperforming white-box attacks

Model	Robust accuracy	
	SOTA	**Square**
Madry et al.	**88.13%**	88.25%
TRADES	93.33%	**92.58%**

Table 3.5: l_2-robustness of the l_∞-adversarially trained models of Madry et al. (2018) at different thresholds ϵ. PGD is shown with 1, 10, 100 random restarts. The black-box attacks are given a 10k queries budget (see the supplement for details)

	Robust accuracy						
ϵ_2	White-box			Black-box			
	PGD_1	PGD_{10}	PGD_{100}	SignHunter	Bandits	SimBA	**Square**
2.0	79.6%	67.4%	**59.8%**	95.9%	80.1%	87.6%	**16.7%**
2.5	69.2%	51.3%	**36.0%**	94.9%	32.4%	75.8%	**2.4%**
3.0	57.6%	29.8%	**12.7%**	93.8%	12.5%	58.1%	**0.6%**

Outperforming white-box attacks on robust models. The models obtained with the adversarial training of Madry et al. (2018) and TRADES (Zhang et al., 2019b) are standard benchmarks to test adversarial attacks, which means that many papers have tried to reduce their robust accuracy, without limit on the computational budget and primarily via white-box attacks. We test our l_∞-Square Attack on these robust models on MNIST at $\epsilon = 0.3$, using $p = 0.8$, 20k queries and 50 random restarts, i.e., we run our attack 50 times and consider it successful if any of the runs finds an adversarial example (Table 3.4). On the model of Madry et al (Madry et al., 2018) Square Attack is only 0.12% far from the *white-box* state-of-the-art, achieving the second best result (also outperforming the 91.47% of SignHunter (Al-Dujaili and O'Reilly, 2020) by a large margin). On the TRADES benchmark (Zheng et al., 2019), our method obtains a new SOTA of 92.58% robust accuracy outperforming the white-box attack of Croce and Hein (2020a). Additionally, the subsequent work of Croce and Hein (2020b) uses the Square Attack as part of their *AutoAttack* where they show that the Square Attack outperforms other white-box attacks on 9 out of 9 MNIST models they evaluated. Thus, our black-box attack can be also useful for robustness evaluation of new defenses in the setting where gradient-based attacks require many restarts and iterations.

Resistance to gradient masking. In Table 3.5 we report the robust accuracy at different thresholds ϵ of the l_∞-adversarially trained models on MNIST of Madry et al. (2018) for the l_2-threat model. It is known that the PGD is ineffective since it suffers from gradient masking (Tramèr and Boneh, 2019). Unlike PGD and other black-box attacks, our Square Attack does not suffer from gradient masking and yields robust accuracy close to zero for $\epsilon = 2.5$, with only a single run. Moreover, the l_2-version of SignHunter (Al-Dujaili and O'Reilly, 2020) fails to accurately assess the robustness because the method optimizes only over the extreme points of the l_∞-ball of radius $\epsilon/\sqrt{d}$ embedded in the target l_2-ball.

Attacking Clean Logit Pairing and Logit Squeezing. These two l_∞ defenses proposed in Kannan et al. (2018) were broken in Mosbach et al. (2018). However, Mosbach et al. (2018) needed up to 10k restarts of PGD which is computationally prohibitive. Using the publicly available models from Mosbach et al. (2018), we run the Square Attack with $p = 0.3$ and 20k query limit (results in Table 3.6). We obtain robust accuracy similar to PGD_R in most cases, but with a *single run*, i.e. without additional restarts. At the same time, although on some models Bandits and SignHunter outperform PGD_1, they on average achieve significantly worse results than the Square Attack. This again shows the

Table 3.6: l_∞-robustness of Clean Logit Pairing (CLP), Logit Squeezing (LSQ) (Kannan et al., 2018). The Square Attack is competitive to white-box PGD with many restarts (R=10,000, R=100 on MNIST, CIFAR-10 resp.) and more effective than black-box attacks (Ilyas et al., 2019a; Al-Dujaili and O'Reilly, 2020)

ϵ_∞	Model	Robust accuracy				
		White-box		Black-box		
		PGD_1	PGD_R	Bandits	SignHunter	**Square**
0.3	CLP_{MNIST}	62.4%	**4.1%**	33.3%	62.1%	**6.1%**
	LSQ_{MNIST}	70.6%	**5.0%**	37.3%	65.7%	**2.6%**
16/255	CLP_{CIFAR}	2.8%	**0.0%**	14.3%	**0.1%**	0.2%
	LSQ_{CIFAR}	27.0%	**1.7%**	27.7%	13.2%	**7.2%**

utility of the Square Attack to accurately assess robustness.

3.7 Conclusion

We have presented a simple black-box attack which outperforms by a large margin the state-of-the-art both in terms of query efficiency and success rate. Our results suggest that our attack is useful *even in comparison to white-box attacks* to better estimate the robustness of models that exhibit gradient masking.

Appendix

3.8 Proofs Omitted from Section 3.4 and Section 3.5

In this section, we present the proofs omitted from Section 3.4 and Section 3.5.

3.8.1 Proof of Proposition 3.4.1

Let δ be the output of Algorithm 3. We prove here that $\|\hat{x} + \delta - x\|_2 = \epsilon$.

From Step 13 of Algorithm 3, we directly have the equality $\|\hat{x} + \delta - x\|_2 = \|\nu\|_2$. Let ν^{old} be the update at the previous iteration, defined in Step 1 and $\overline{W_1 \cup W_2}$ the indices not belonging to $W_1 \cup W_2$. Then,

$$\begin{aligned}
\|\nu\|_2^2 &= \sum_{i=1}^{c} \|\nu_{W_1 \cup W_2, i}\|_2^2 + \sum_{i=1}^{c} \left\|\nu_{\overline{W_1 \cup W_2}, i}\right\|_2^2 \\
&= \sum_{i=1}^{c} \|\nu_{W_1, i}\|_2^2 + \sum_{i=1}^{c} \left\|\nu_{\overline{W_1 \cup W_2}, i}\right\|_2^2 \\
&= \sum_{i=1}^{c} (\epsilon_{\text{avail}}^i)^2 + \sum_{i=1}^{c} \left\|\nu_{\overline{W_1 \cup W_2}, i}\right\|_2^2 \\
&= \sum_{i=1}^{c} \left\|\nu^{\text{old}}_{W_1 \cup W_2, i}\right\|_2^2 + \epsilon_{\text{unused}}^2 + \sum_{i=1}^{c} \left\|\nu_{\overline{W_1 \cup W_2}, i}\right\|_2^2 \\
&\overset{(i)}{=} \sum_{i=1}^{c} \left\|\nu^{\text{old}}_{W_1 \cup W_2, i}\right\|_2^2 + \epsilon_{\text{unused}}^2 + \sum_{i=1}^{c} \left\|\nu^{\text{old}}_{\overline{W_1 \cup W_2}, i}\right\|_2^2 \\
&= \left\|\nu^{\text{old}}\right\|_2^2 + \epsilon_{\text{unused}}^2 \overset{(ii)}{=} \epsilon^2,
\end{aligned}$$

where (i) holds since $\nu^{\text{old}}_{\overline{W_1 \cup W_2}} \equiv \nu_{\overline{W_1 \cup W_2}}$ as the modifications affect only the elements in the two windows, and (ii) holds by the definition of ϵ_{unused} in Step 4 of Algorithm 3.

3.8.2 Proof of Proposition 3.5.1

Using the L-smoothness of the function g, that is it holds for all $x, y \in \mathbb{R}^d$,

$$\|\nabla g(x) - \nabla g(y)\|_2 \leq L \|x - y\|_2 .$$

we obtain (see e.g. Boyd and Vandenberghe (2004)):

$$g(x_t + \delta_t) \leq g(x_t) + \langle \nabla g(x_t), \delta_t \rangle + \frac{L}{2} \|\delta_t\|_2^2,$$

and by definition of x_{t+1} we have

$$\begin{aligned} g(x_{t+1}) &\leq \min\{g(x_t), g(x_t + \delta_t)\} \\ &\leq g(x_t) + \min\{0, \langle \nabla g(x_t), \delta_t \rangle + \frac{L}{2}\|\delta_t\|_2^2\}. \end{aligned}$$

Using the definition of the min as a function of the absolute value ($2\min\{a, b\} = a + b - |a - b|$) yields

$$g(x_{t+1}) \leq g(x_t) + \frac{1}{2}\langle \nabla g(x_t), \delta_t \rangle + \frac{L}{4}\|\delta_t\|_2^2 - \frac{1}{2}|\langle \nabla g(x_t), \delta_t \rangle + \frac{L}{2}\|\delta_t\|_2^2|.$$

And using the triangular inequality ($|a + b| \geq |a| - |b|$), we have

$$g(x_{t+1}) \leq g(x_t) + \frac{1}{2}\langle \nabla g(x_t), \delta_t \rangle + \frac{L}{2}\|\delta_t\|_2^2 - \frac{1}{2}|\langle \nabla g(x_t), \delta_t \rangle|.$$

Therefore taking the expectation and using that $\mathbb{E}\delta_t = 0$, we get

$$\mathbb{E}g(x_{t+1}) \leq \mathbb{E}g(x_t) - \frac{1}{2}\mathbb{E}|\langle \nabla g(x_t), \delta_t \rangle| + \frac{L}{2}\mathbb{E}\|\delta_t\|_2^2.$$

Therefore, together with the assumptions in Eq. (3.3) this yields to

$$\mathbb{E}g(x_{t+1}) \leq \mathbb{E}g(x_t) - \frac{\tilde{C}\gamma_t}{2}\mathbb{E}\|\nabla g(x_t)\|_2 + \frac{LC\gamma_t^2}{2}.$$

and thus

$$\mathbb{E}\,\|\nabla g(x_t)\|_2 \leq \frac{2}{\gamma_t \tilde{C}}\left(\mathbb{E}g(x_t) - \mathbb{E}g(x_{t+1}) + \frac{LC\gamma_t^2}{2}\right).$$

Thus for $\gamma_t = \gamma$ we have summing for $t = 0 : T$

$$\begin{aligned} \min_{0 \leq i \leq T} \mathbb{E}\|\nabla g(x_i)\|_2 &\leq \frac{1}{T}\sum_{t=0}^{T} \mathbb{E}\|\nabla g(x_t)\|_2 \\ &\leq \frac{2}{\tilde{C}\gamma T}\Big[g(x_0) - \mathbb{E}g(x_{T+1}) + \frac{TLC\gamma^2}{2}\Big]. \end{aligned}$$

We conclude setting the step-size to $\gamma = \Theta(1/\sqrt{T})$.

3.8.3 Assumptions in Eq. (3.3) Do Not Hold for the Sampling Distribution P

Let us consider an update δ with a window size $h = 2$ and the direction $v \in \{-1, 1\}^{w \times w \times c}$ defined as

$$v_{k,l}^i = (-1)^{kl} \quad \text{for all } i, k, l.$$

It is easy to check that any update δ drawn from the sampling distribution P is orthogonal to this direction v:

$$\langle v, \delta \rangle = \sum_{i=1}^{c} \sum_{k=r+1}^{r+2} \sum_{l=s+1}^{s+2} (-1)^{kl} = c(-1+1-1+1) = 0.$$

Thus, $\mathbb{E}|\langle v, \delta \rangle| = 0$ and the assumptions in Eq. (3.3) do not hold. This means that the convergence analysis does not directly hold for the sampling distribution P.

3.8.4 Assumptions in Eq. (3.3) Hold for the Sampling Distribution P^{multiple}

Let us consider the sampling distribution P^{multiple} where different Rademacher $\rho_{k,l,i}$ are drawn for each pixel of the update window $\delta_{r+1:r+h,\, s+1:s+h,\, i}$. We present it in Algorithm 4 with the convention that any subscript $k > w$ should be understood as $k-w$. This technical modification is greatly helpful to avoid side effect.

Let $v \in \mathbb{R}^{w \times w \times c}$ for which we have using the Khintchine inequality (Haagerup, 1981):

$$\begin{aligned}
\mathbb{E}|\langle \delta, v \rangle| &= \mathbb{E}| \sum_{k=r+1}^{r+h} \sum_{l=s+1}^{s+h} \sum_{i=1}^{c} \delta_{k,l}^{i} v_{k,l}^{i}| \\
&\overset{(i)}{=} \mathbb{E}_{(r,s)} \mathbb{E}_{\rho} | \sum_{k=r+1}^{r+h} \sum_{l=s+1}^{s+h} \sum_{i=1}^{c} \delta_{k,l}^{i} v_{k,l}^{i}| \\
&\overset{(ii)}{\geq} \frac{2\varepsilon}{\sqrt{2}} \mathbb{E}_{(r,s)} \|V_{(r,s)}\|_2 \\
&\overset{(iii)}{\geq} \sqrt{2}\varepsilon \|\mathbb{E}_{(r,s)} V_{(r,s)}\|_2 \\
&\geq \frac{\sqrt{2}\varepsilon h^2}{w^2} \|v\|_2,
\end{aligned}$$

where we define by $V_{(r,s)} = \{v_{k,l}^{i}\}_{k \in \{r+1,\ldots,r+h\}, l \in \{s+1,\ldots,s+h\}, i \in \{1,\ldots,c\}}$ and (i) follows from the decomposition between the randomness of the Rademacher and the random window, (ii) follows from the Khintchine inequality and (iii) follows from Jensen inequality.

Algorithm 4: Sampling distribution P^{multiple} for l_∞-norm

Input: maximal norm ϵ, window size h, image size w, color channels c
Output: New update δ

$\delta \leftarrow$ array of zeros of size $w \times w \times c$
sample uniformly $r, s \in \{0, \ldots, w\} \subset \mathbb{N}$
for $i = 1, \ldots, c$ **do**
  $\delta_{r+1:r+h,\, s+1:s+h,\, i} \leftarrow Uniform(\{-2\epsilon, 2\epsilon\}^{h \times h})$
end

In addition we have for the variance:

$$\begin{aligned}\mathbb{E}\|\delta\|_2^2 &= \mathbb{E}_{(r,s)} \sum_{k=r+1}^{r+h} \sum_{l=s+1}^{s+h} \sum_{i=1}^{c} \mathbb{E}_\rho (\delta_{k,l}^i)^2 \\ &= \mathbb{E}_{(r,s)} \sum_{k=r+1}^{r+h} \sum_{l=s+1}^{s+h} \sum_{i=1}^{c} 4\varepsilon^2 \\ &= 4c\varepsilon^2 h^2.\end{aligned}$$

Thus the assumptions in Eq. (3.3) hold for the sampling distribution P^{multiple}.

3.8.5 Why Updates of Equal Sign?

Proposition 3.5.1 underlines the importance of a large inner product $\mathbb{E}[|\langle \delta_t, \nabla g(x_t)\rangle|]$ in the direction of the gradients. This provides some intuition explaining why the update δ^{single}, where a single Rademacher is drawn for each window, is more efficient than the update δ^{multiple} where different Rademacher values are drawn. Following the observation that the gradients are often approximately piecewise constant (Ilyas et al., 2019a), we consider, as a heuristic, a piecewise constant direction v for which we will show that

$$\mathbb{E}[|\langle \delta^{\text{single}}, v\rangle|] = \Theta(\|v\|_1) \quad \text{and} \quad \mathbb{E}[|\langle \delta^{\text{multiple}}, v\rangle|] = \Theta(\|v\|_2).$$

Therefore the directions sampled by our proposal are more correlated with the gradient direction and help the algorithm to converge faster. This is also verified empirically in our experiments (see the ablation study in Sup. 3.10).

Analysis. Let us consider the direction $v \in \mathbb{R}^{w \times w}$ composed of different blocks

$$\{V_{(r,s)}\}_{(r,s) \in \{0,\dots,w/h\}}$$

of constant sign.

For this direction v we compare two different proposal P^{multiple} and P^{single} where we choose uniformly one random block (r, s) and we either assign a single Rademacher $\rho_{(r,s)}$ to the whole block (this is P^{single}) or we assign multiple Rademacher values

$$\{\rho_{(k,l)}\}_{k \in \{rh+1,\dots,(r+1)h\}, l \in \{sh+1,\dots,(s+1)h\}}$$

(this is P^{multiple}). Using the Khintchine and Jensen inequalities similarly to Sec. 3.8.4, we have

$$\begin{aligned}\mathbb{E}|\langle \delta^{\text{multiple}}, v\rangle| &= \mathbb{E}|\sum_{k=rh+1}^{(r+1)h} \sum_{l=sh+1}^{(s+1)h} \delta_{k,l} v_{k,l}| \\ &\geq \frac{2\varepsilon}{\sqrt{2}} \mathbb{E}_{(r,s)} \|V_{(r,s)}\|_2\end{aligned}$$

$$\geq \frac{\sqrt{2}\varepsilon h^2}{w^2}\|v\|_2.$$

Moreover, we can show the following upper bound using the Khintchine inequality and the inequality between the l_1- and l_2-norms:

$$\begin{aligned}
\mathbb{E}|\langle \delta^{\text{multiple}}, v\rangle| &= \mathbb{E}|\sum_{k=rh+1}^{(r+1)h}\sum_{l=sh+1}^{(s+1)h}\delta_{k,l}v_{k,l}| \\
&\leq 2\varepsilon\mathbb{E}_{(r,s)}\|V_{(r,s)}\|_2 \\
&= \frac{2\varepsilon h^2}{w^2}\sum_{r=1}^{w/h}\sum_{s=1}^{w/h}\|V_{(r,s)}\|_2 \\
&\leq \frac{2\varepsilon h}{w}\|v\|_2
\end{aligned}$$

Thus, $\mathbb{E}[|\langle \delta^{\text{multiple}}, v\rangle|] = \Theta(\|v\|_2)$.

For the update δ^{single} we obtain

$$\begin{aligned}
\mathbb{E}|\langle \delta^{\text{single}}, v\rangle| &= \mathbb{E}|\sum_{k=rh+1}^{(r+1)h}\sum_{l=sh+1}^{(s+1)h}\delta_{r,s}v_{k,l}| \\
&= \mathbb{E}|\delta_{r,s}\sum_{k=rh+1}^{(r+1)h}\sum_{l=sh+1}^{(s+1)h}v_{k,l}| \\
&\overset{(i)}{=} 2\varepsilon\mathbb{E}_{(r,s)}\|V_{(r,s)}\|_1 \\
&= \frac{2\varepsilon h^2}{w^2}\|v\|_1
\end{aligned}$$

where (i) follows from the fact the $V_{(r,s)}$ has a constant sign. We recover then the l_1-norm of the direction v, i.e. we conclude that $\mathbb{E}[|\langle \delta^{\text{single}}, v\rangle|] = \Theta(\|v\|_1)$.

This implies that for an approximately constant block $\mathbb{E}|\langle \delta^{\text{single}}, v\rangle|$ will be larger than $\mathbb{E}|\langle \delta^{\text{multiple}}, v\rangle|$. For example, in the extreme case of constant binary block $|V_{(r,s)}| = 11^\top$, we have

$$\mathbb{E}|\langle \delta^{\text{single}}, v\rangle| = 2\varepsilon h^2 >> \mathbb{E}|\langle \delta^{\text{multiple}}, v\rangle| \asymp 2\varepsilon h.$$

3.8.6 Proof of Proposition 3.5.2

Let $x \in \mathcal{F}$, and $N(x)$ the number of elements of $\mathcal{G}$ that x contains. Let initialize x as a square of size $s \times s$, so that $N(x) = 1$. We then add iteratively the remaining $k - s^2$ unitary squares to x so to maximize $N(x)$.
In order to get $N(x) = 2$ it is necessary to increase x to have size $s \times (s+1)$. At this point, again to get $N(x) = 3$ we need to add s squares to one side of x. However, if we choose to glue them so to form a rectangle $s \times (s+2)$, then $N(x) = 3$ and once more we need other s squares to increase N, which means overall $s^2 + 3s$ to achieve $N(x) = 4$. If instead

we glue s squares along the longer side, with only one additional unitary square we get $N(x) = 4$ using $s^2+2s+1 < s^2+3s$ unitary squares (as $s \geq 2$), with $x = (s+1) \times (s+1)$. Then, if the current shape of x is $a \times b$ with $a \geq b$, the optimal option is adding a unitary squares to have shape $a \times (b+1)$, increasing the count N of $a - s + 1$. This strategy can be repeated until the budget of k unitary squares is reached.
Finally, since we start from the shape $s \times s$, then at each stage $b - a \in \{0, 1\}$, which means that the final a will be $\left\lfloor \sqrt{k} \right\rfloor$. A rectangle $a \times b$ in $\mathcal{F}$ contains $(a-s+1)(b-s+1)$ elements of $\mathcal{G}$. The remaining $k - ab$ squares can be glued along the longer side, contributing to $N(x)$ with $(k - ab - s + 1)^+$.

3.9 Experimental Details

In this section, we list the main hyperparameters and various implementation details for the experiments done in the main experiments (Sec. 3.6).

3.9.1 Experiments on ImageNet

For the untargeted Square Attack on the ImageNet models, we used $p = 0.05$ and $p = 0.1$ for the l_∞- and l_2- versions respectively. For Bandits, we used their code with their suggested hyperparameters (specified in the configuration files) for both l_∞ and l_2. For SignHunter, we used directly their code which does not have any hyperparameters (assuming that the finite difference probe δ is set to ϵ). For SimBA-DCT, we used the default parameters of the original code apart from the following, which are the suggested ones for each model: for ResNet-50 and VGG-16-BN "freq_dims" = 28, "order" = "strided" and "stride" = 7, for Inception v3 "freq_dims" = 38, "order" = "strided" and "stride" = 9. Notice that SimBA tries to minimize the l_2-norm of the perturbations but it does not have a bound on the size of the changes. Then we consider it successful when the adversarial examples produced have norm smaller than the fixed threshold ϵ. The results for all other methods were taken directly from the corresponding papers.

Evaluation of Bandits. The code of Bandits (Ilyas et al., 2019a) does not have image standardization at the stage where the set of correctly points is determined (see `https://github.com/MadryLab/blackbox-bandits/issues/3`). As a result, the attack is run only on the set of points correctly classified by the network *without standardization*, although the network was trained on standardized images. We fix this bug, and report the results in Table 3.2 based on the fixed version of their code. We note that the largest difference of our evaluation compared to the l_∞ results reported in Appendix E of Ilyas et al. (2019a) is obtained for the VGG-16-BN network: we get 2.0% failure rate while they reported 8.4% in their paper. Also, we note that the query count for Inception v3 we obtain is also better than reported in Ilyas et al. (2019a): 957 instead of 1117 with a slightly better failure rate. Our l_2 results also differ – we obtain a significantly lower failure rate (9.8%, 6.8%, 10.2% instead of 15.5%, 9.7%, 17.2% for the Inception v3, ResNet-50, VGG-16-BN networks respectively) with improved average number of queries (1486, 939, 511 instead

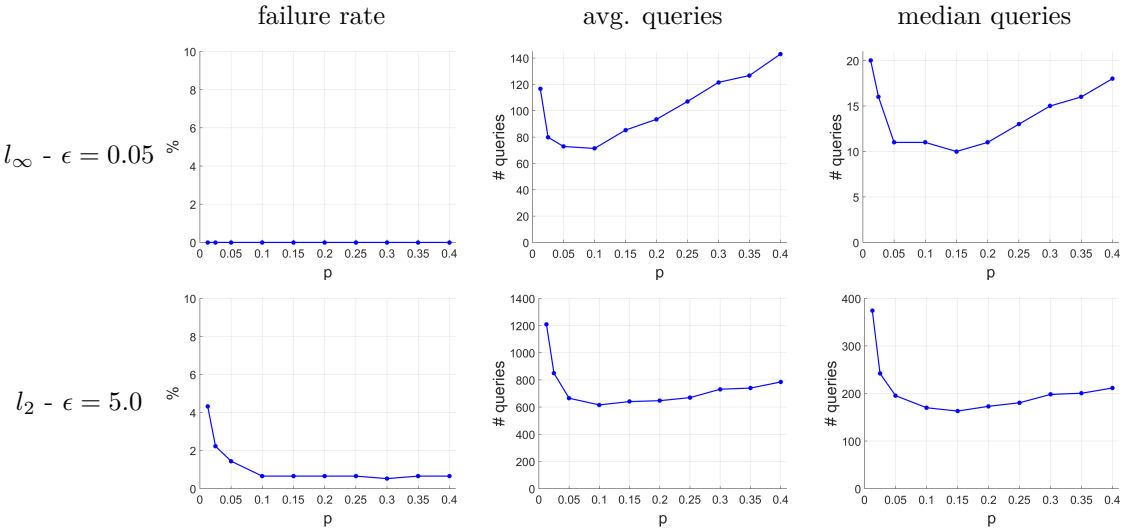


Figure 3.5: Sensitivity of the Square Attack to different choices of $p \in \{0.0125, 0.025, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4\}$, i.e. the initial fraction of pixels changed by the attack, on ImageNet for a ResNet-50 model

of 1858, 993, 594).

3.9.2 Square Attack Can be More Accurate than White-box Attacks

For the l_∞-Square Attack, we used $p = 0.3$ for all models on MNIST and CIFAR-10. For Bandits on MNIST and CIFAR-10 adversarially trained models we used "exploration" = 0.1, "tile size" = 16, "gradient iters" = 1 following (Seungyong et al., 2019).

For the comparison of l_2-attacks on the l_∞-adversarially trained model of Madry et al. (2018) we used the Square Attack with the usual parameter $p = 0.1$. For Bandits we used the parameters "exploration" = 0.01, "tile size" = 28, "gradient iters" = 1, after running a grid search over the three of them (all the other parameters are kept as set in the original code). For SimBA we used the "pixel attack" with parameters "order" = "rand", "freq_dims" = 28, step size of 0.50, after a grid search on all the parameters.

3.10 Ablation Study

Here we discuss in more detail the ablation study which justifies the algorithmic choices made for our l_∞- and l_2-attacks. Additionally, we discuss the robustness of the attack to the hyperparameter p, i.e. the initial fraction of pixels changed by the attack (see Fig. 3.5). We perform all these experiments on ImageNet with a standardly trained ResNet-50 model from the PyTorch repository.

3.10.1 l_∞-Square Attack

Sensitivity to the hyperparameter p. First of all, we note that for *all* values of p we achieve 0.0% failure rate. Moreover, we achieve state-of-the-art query efficiency with *all* considered values of p (from 0.0125 to 0.4), i.e. we have the average number of queries below 140, and the median below 20 queries. Therefore, we conclude that the attack is robust to a wide range of p, which is an important property of a black-box attack – since the target model is unknown, and one aims at minimizing the number of queries needed to fool the model, doing even an approximate grid search over p is prohibitively expensive.

Algorithmic choices. In addition to the results presented in Sec. 3.5.3, we show in Table 3.7 the results of a few more variants of the Square Attack. We recall that "# random signs" indicates how many different signs we sample to build the updates, with c being the number of color channels and h the current size of the square-shaped updates. Specifically, we test the performance of using a single random sign for all the elements for the update, "square-1", which turns out to be comparable to "square-$c \cdot h^2$", i.e. every component of the update has sign independently sampled, but worse than keeping the sign constant within each color channel ("square-c").

In order to implement update shape "rectangle", on every iteration and for every image we sample $\alpha, \beta \sim \text{Exp}(1)$ and take a rectangle with sides $\alpha \cdot s$ and $\beta \cdot s$, so that in expectation its area is equal to s^2, i.e. to the area of the original square. This update scheme performs significantly better than changing a random subset of pixels (93 vs 339 queries on average), but worse than changing squares (73 queries on average) as discussed in Sec. 3.5.2.

Finally, we show the results with two more initialization schemes: horizontal stripes (instead of vertical), as well as initialization with randomly placed squares. While both solutions lead to the state-of-the-art query efficiency (83 and 90 queries on average) compared to the literature, they achieve worse results than the vertical stripes we choose for our Square Attack.

3.10.2 l_2-Square Attack

Sensitivity to the hyperparameter p. We observe that the l_2-Square Attack is robust to different choices in the range between 0.05 and 0.4 showing approximately the same failure rate and query efficiency for all values of p in this range, while its performance degrades slightly for very small initial squares $p \in \{0.0125, 0.025\}$.

Algorithmic choices. We analyze in Table 3.7 the sensitivity of the l_2-attack to different choices of the shape of the update and initialization.

In particular, we test an update with only one "center" instead of two, namely $\eta^{\text{single}} = \eta^{h,h}$ (following the notation of Eq. 3.2) and one, η^{rand}, where the step 7 in Algorithm 3 is $\rho \leftarrow Uniform(\{-1, 1\}^{h \times h})$ instead of $\rho \leftarrow Uniform(\{-1, 1\})$, which means that each element of η is multiplied randomly by either -1 or 1 independently (instead of all elements

Table 3.7: An ablation study for the performance of the l_∞- and l_2-Square Attack under various algorithmic choices of the attack. The metrics are calculated on 1,000 ImageNet images for a ResNet-50 model. The last row represents our recommended setting. For all experiments we used the best performing p (0.05 for l_∞ and 0.1 for l_2)

l_∞ ablation study

Update shape	# random signs	Initialization	Failure rate	Avg. queries	Median queries
random	$c \cdot h^2$	vert. stripes	0.0%	401	48
random	$c \cdot h^2$	uniform rand.	0.0%	393	132
random	c	vert. stripes	0.0%	339	53
square	$c \cdot h^2$	vert. stripes	0.0%	153	15
square	1	vert. stripes	0.0%	129	18
rectangle	c	vert. stripes	0.0%	93	16
square	c	uniform rand.	0.0%	91	26
square	c	rand. squares	0.0%	90	20
square	c	horiz. stripes	0.0%	83	18
square	c	vert. stripes	0.0%	73	11

l_2 ablation study

Update	Initialization	Failure rate	Avg. queries	Median queries
η^{rand}	η^{rand}-grid	3.3%	1050	324
η^{single}	η^{single}-grid	0.7%	650	171
η	gaussian	0.4%	696	189
η	uniform	0.8%	660	187
η	vert. stripes	0.8%	655	186
η	η-grid	0.7%	616	170

multiplied by the same value). We can see that using different random signs in the update and initialization (η^{rand}) significantly (1.5× factor) degrades the results for the l_2-attack, which is similar to the observation made for the l_∞-attack.

Alternatively to the grid described in Sec. 3.4.4, we consider as starting perturbation i) a random point sampled according to $Uniform(\{-\epsilon/\sqrt{d}, \epsilon/\sqrt{d}\}^{w \times w \times c})$, that is on the corners of the largest l_∞-ball contained in the l_2-ball of radius ϵ (uniform initialization), ii) a random position on the l_2-ball of radius ϵ (Gaussian initialization) or iii) vertical stripes similarly to what done for the l_∞-Square Attack, but with magnitude $\epsilon/\sqrt{d}$ to fulfill the constraints on the l_2-norm of the perturbation. We note that different initialization schemes do not have a large influence on the results of our l_2-attack, unlike for the l_∞-attack.

3.11 Stability of the Attack under Different Random Seeds

Here we study the stability of the Square Attack over the randomness in the algorithm, i.e. in the initialization, in the choice of the locations of square-shaped regions, and in the choice of the values in the updates δ. We repeat 10 times experiments similar to the ones reported in Sec. 3.6.1 and Sec. 3.6.2 with different random seeds for our attack, and

Table 3.8: Mean and standard deviation of the main performance metrics of the Square Attack across 10 different runs with different random seeds

ImageNet, ResNet-50

Norm	ϵ	Failure rate	Avg. queries	Median queries
l_∞	0.05	$0.0\% \pm 0.0\%$	72 ± 2	11 ± 1
l_2	5	$0.6\% \pm 0.1\%$	638 ± 12	163 ± 8

MNIST, adversarially trained LeNet from (Madry et al., 2018)

Norm	ϵ	Robust accuracy	Avg. queries	Median queries
l_∞	0.3	$87.0\% \pm 0.1\%$	299 ± 47	52 ± 7
l_2	2	$16.0\% \pm 1.4\%$	1454 ± 71	742 ± 78

report all the metrics with standard deviations in Table 3.8. On ImageNet, we evaluate the *failure rate* (over initially correctly classified points) and query efficiency on 1,000 images using ResNet-50. On MNIST, we evaluate the *robust accuracy* (i.e. the failure rate over all points) and query efficiency on 1,000 images using the l_∞-adversarially trained LeNet from Madry et al. (2018). Note that unlike in Sec. 3.6.2 in both cases we use a single restart for the attack on MNIST, and we compute the statistics on 1,000 points instead of 10,000, thus the final results will differ.

On the ImageNet model, all these metrics are very concentrated for both the l_∞- and the l_2-norms. Moreover, we note that the standard deviations are much smaller than the gap between the Square Attack and the competing methods reported in Table 3.2. Thus we conclude that the results of the attack are stable under different random seeds.

On the adversarially trained MNIST model from Madry et al. (2018), the robust accuracy is very concentrated showing only 0.1% and 1.4% standard deviations for the l_∞- and the l_2-norms respectively. Importantly, this is much less than the gaps to the nearest competitors reported in Tables 3.4 and 3.5. We also show query efficiency for this model, although for models with non-trivial robustness it is more important to achieve lower robust accuracy, and query efficiency on successful adversarial examples is secondary. We note that the standard deviation of the mean and median number of queries is higher than for ImageNet, particularly for the l_∞-ball of radius $\epsilon = 0.3$ where the robust accuracy is much higher than for the l_2-ball of radius $\epsilon = 2$. This is possibly due to the fact that attacking more robust models (within a certain threat model) is a more challenging task than, e.g., attacking standardly trained classifiers, as those used on ImageNet, which means that a favorable random initialization or perturbation updates can have more influence on the query efficiency.

3.12 Additional Experimental Results

This section contains results on targeted attacks, and also additional results on untargeted attacks that complement the ImageNet results from Table 3.2. Moreover, we show that the Square Attack is useful for evaluating the robustness of newly proposed defenses (see

Sec. 3.12.6).

3.12.1 Targeted Attacks

While in Sec. 3.6 we considered only untargeted attacks, here we report the results of the different attacks in the *targeted* scenario.

Targeted Square Attack. In order to adapt our scheme to targeted attacks, where one first choose a target class t and then tries to get the model f to classify a point x as t, we need to modify the loss function L which is minimized (see Eq. (3.1)). For the untargeted attacks we used the margin-based loss $L(f(x), y) = f_y(x) - \max_{k \neq y} f_k(x)$, with y the correct class of x. This loss could be straightforwardly adapted to the targeted case as $L(f(x), t) = -f_t(x) + \max_{k \neq t} f_k(x)$. However, in practice we observed that this loss leads to suboptimal query efficiency. We hypothesize that the drawback of the margin-based loss in this setting is that the maximum over $k \neq t$ is realized by different k at different iterations, and then the changes applied to the image tend to cancel each other. We observed this effect particularly on ImageNet which has a very high number of classes.

Instead, we use here as objective function the cross-entropy loss on the target class, defined as

$$L(f(x), t) = -f_t(x) + \log \left(\sum_{i=1}^{K} e^{f_i(x)} \right). \tag{3.5}$$

Minimizing L is then equivalent to maximizing the confidence of the classifier in the target class. Notice that in Eq. (3.5) the scores of all the classes are involved, so that it increases the *relative* weight of the target class respect to the others, making the targeted attacks more effective.

Table 3.9: Results of **targeted** attacks on ImageNet for Inception-v3 model using 100k query limit for l_∞, 60k for l_2. The results for the competing methods for l_∞ are taken from Meunier et al. (2019), except SignHunter (Al-Dujaili and O'Reilly, 2020) which we evaluated using their code

Norm	Attack	Failure rate	Avg. queries	Median queries
l_∞	Bandits	7.5%	25341	18053
	SignHunter	1.1%	8814	5481
	Parsimonious	**0.0%**	7184	5116
	DFO_c – Diag. CMA	6.0%	6768	3797
	DFO_c – CMA	**0.0%**	6662	4692
	Square Attack	**0.0%**	**4584**	**2859**
l_2	Bandits	**24.5%**	20489	17122
	SimBA-DCT	25.5%	30576	30180
	Square Attack	33.5%	**19794**	**15946**

Experiments. We present the results for targeted attacks on ImageNet for Inception-v3 model in Table 3.9. We calculate the statistics on 1,000 images (the target class is randomly picked for each image) with query limit of 100,000 for l_∞ and on 200 points

and with query limit 60,000 for l_2, as this one is more expensive computationally because of the lower success rate, with the same norm bounds ϵ used in the untargeted case. We use the Square Attack with $p = 0.01$ for l_∞ and $p = 0.02$ for l_2. The results for the competing methods for l_∞ are taken from Meunier et al. (2019), except SignHunter (Al-Dujaili and O'Reilly, 2020) which was not evaluated in the targeted setting before, thus we performed the evaluation using their code on 100 test points using the cross-entropy as the loss function. For l_2, we use Bandits (Ilyas et al., 2019a) with the standard parameters used in the untargeted scenario, while we ran a grid search over the step size of SimBA (we set it to 0.03) and keep the other hyperparameters as suggested for the Inception-v3 model.

The targeted l_∞-Square Attack achieves 100% success rate and requires 1.5 times fewer queries on average than the nearest competitor (Meunier et al., 2019), showing that even in the *targeted* scenario our simple scheme outperforms the state-of-the-art methods. On the other hand, our l_2-attack suffers from worse higher failure rate than the competitors, but achieves lower average and median number of queries (although on a different number of successful points).

3.12.2 Success Rate on ImageNet for Different Number of Queries

In this section, we provide a more detailed comparison to the competitors from Table 3.2 under different query budgets and more comments about the low query regime experiment in Fig. 3.4. We show in Fig. 3.6 the behaviour of the success rate for each attack depending on the number queries. The success rates of the attacks from Meunier et al. (2019) (DFO_c–CMA–50 and DFO_d–Diag. CMA–30) and Shukla et al. (2019) (BayesAttack) for different number of queries were obtained via personal communication directly from the authors, and were calculated on 500 and 10,000 randomly sampled points, respectively. For the other attacks, as mentioned above, the success rate is calculated on 1,000 randomly sampled points.

l_∞-results. First, we observe that the Square Attack outperforms all other methods in the standard regime with 10,000 queries. The gap in the success rate gets larger in the range of 100-1000 queries for the more challenging Inception-v3 model, where we observe over 10% improvement in the success rate over all other methods including SignHunter. Our method also outperforms the BayesAttack in the low query regime, i.e. less than 200 queries, by approximately 20% on every model. We note that DFO_d–Diag. CMA–30 method is also quite effective in the low query regime showing results close to BayesAttack. However, it is also outperformed by our Square Attack.

l_2-results. First, since the l_2-version of SignHunter (Al-Dujaili and O'Reilly, 2020) is not competitive to Bandits on ImageNet (see Fig. 2 in Al-Dujaili and O'Reilly (2020)), we do not compare to them here. The l_2-Square Attack outperforms both Bandits and SimBA, and the gap is particularly large in the low query regime. We note that the success rate of SimBA plateaus after some iteration. This happens due to the fact that their algorithm

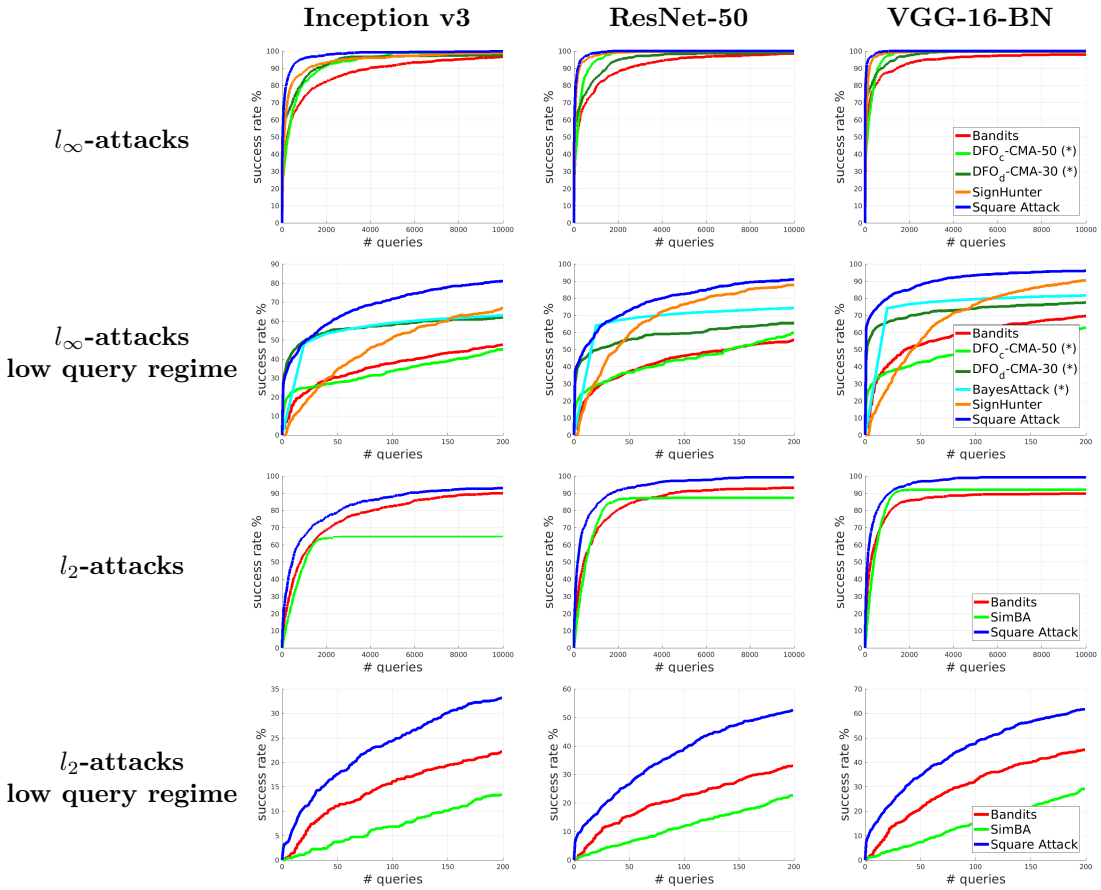


Figure 3.6: Success rate vs number of queries for different attacks on ImageNet on three standardly trained models. The low query regime corresponds to up to 200 queries, while the standard regime corresponds to 10,000 queries. * denotes the results obtained via personal communication with the authors and evaluated on 500 and 10,000 randomly sampled points for DFO (Meunier et al., 2019) and BayesAttack (Shukla et al., 2019) methods, respectively

only adds orthogonal updates to the perturbation, and does not have any way to correct the greedy decisions made earlier. Thus, there is no progress anymore after the norm of the perturbation reaches the $\epsilon = 5$ (note that we used for SimBA the same parameters of the comparison between SimBA and Bandits in Guo et al. (2019b)). Contrary to this, both Bandits and our attack constantly keep improving the success rate, although with a different speed.

3.12.3 Performance on Architectures with Dilated Convolutions

In Sec. 3.5.2, we provided justifications for square-shaped updates for *convolutional* networks. Thus, a reasonable question is whether the Square Attack still works equally well on less standard convolutional networks such as, for example, networks with dilated convolutions. For this purpose, we evaluate three different architectures introduced in Yu et al. (2017) that involve *dilated* convolutions: DRN-A-50, DRN-C-42 and DRN-D-38. We use $\epsilon_\infty = 0.05$ as in the main ImageNet experiments from Table 3.2. We present the results

Table 3.10: Results of untargeted l_∞-perturbations produced by the Square Attack on architectures with dilated convolutions

ϵ_∞	Model	Failure rate	Avg. queries	Median queries
0.05	DRN-A-50	0.0%	86	12
	DRN-C-42	0.0%	57	7
	DRN-D-38	0.0%	48	6

in Table 3.10 and observe that for all the three model the Square Attack achieves 100% success rate and both average and median number of queries stay comparable to that of VGG or ResNet-50 from Table 3.2. Thus, this experiment suggests that our attack can be applied not only to standard convolutional networks, but also to more recent neural network architectures.

3.12.4 Imperceptible Adversarial Examples with the Square Attack

Adversarial examples in general need not be imperceptible, for example adversarial patches (Brown et al., 2017; Karmon et al., 2018) are clearly visible, and yet can be used to attack machine learning systems deployed in-the-wild. However, if imperceptibility is the goal, it can be easily ensured by adjusting the size of the allowed perturbations. In the main ImageNet experiments in Table 3.2 we used $\epsilon_\infty = 0.05 = 12.75/255$ since this is standard in the literature (Al-Dujaili and O'Reilly, 2020; Ilyas et al., 2019a; Meunier et al., 2019; Seungyong et al., 2019), and for which all the considered attacks produce visible perturbations. Below we additionally provide results on the more than $3\times$ smaller threshold $\epsilon_\infty = 4/255$ which leads to imperceptible perturbations (see Fig. 3.7). Our attack still achieves almost perfect success rate requiring only a limited number of queries as shown in Table 3.11. Thus, one can also generate imperceptibile adversarial examples with the Square Attack simply by adjusting the perturbation size.

3.12.5 Analysis of Adversarial Examples that Require More Queries

Here we provide more visualizations of adversarial perturbations generated by the untargeted Square Attack for $\epsilon = 0.05$ on ImageNet. We analyze here the inputs that require more queries to be misclassified. We present the results in Fig. 3.8 where we plot adversarial examples after 10, 100 and 500 iterations of our attack. First, we note that a misclassification is achieved when the margin loss becomes negative. We can observe that the loss decreases gradually over iterations, and a single update only rarely leads to a

Table 3.11: Results of untargeted imperceptible l_∞-perturbations produced by the Square Attack on standard architectures

ϵ_∞	Model	Failure rate	Avg. queries	Median queries
4/255	VGG	0.5%	424	115
	ResNet-50	0.3%	652	213
	Inception v3	5.4%	1013	391

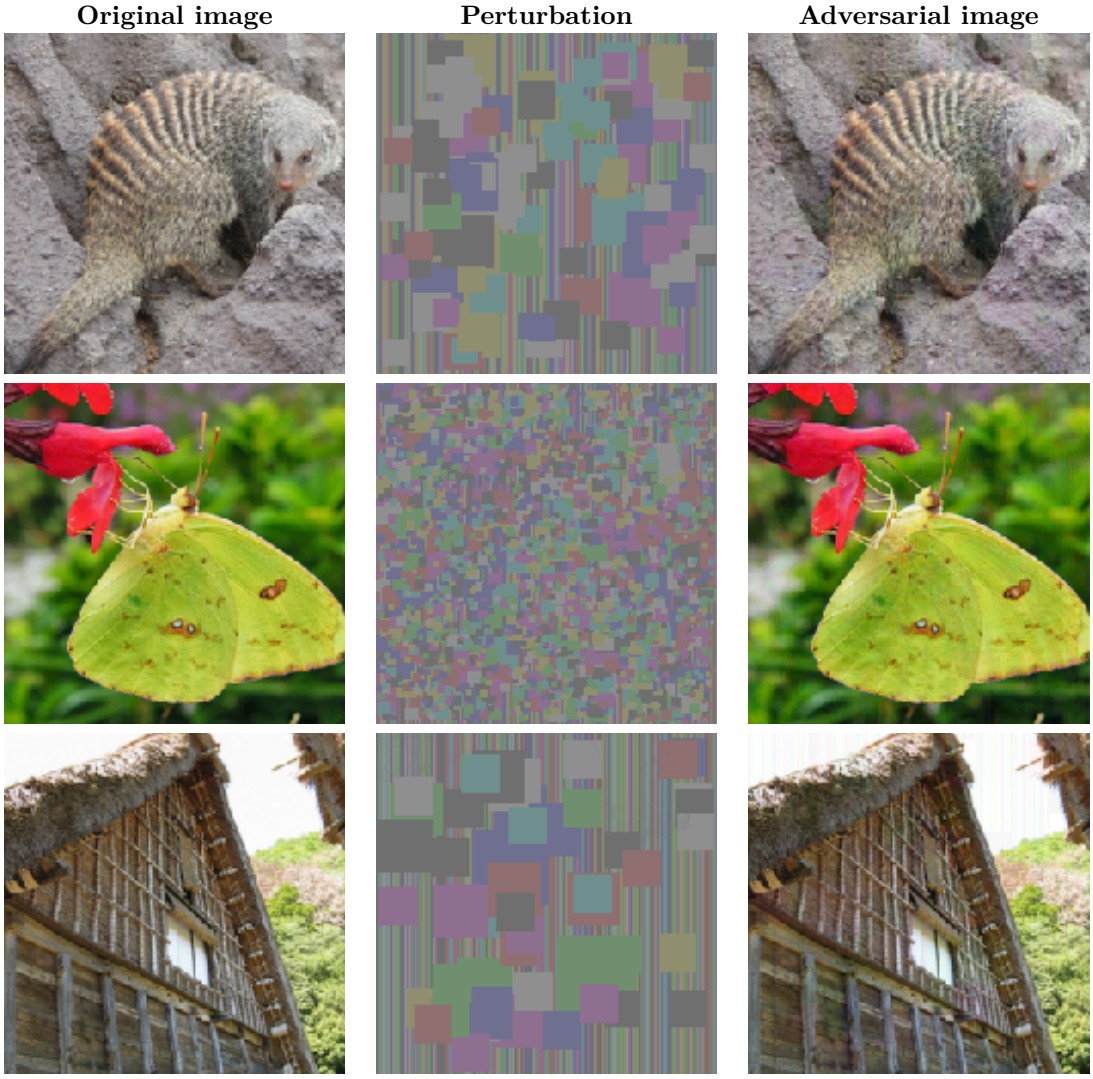


Figure 3.7: Visualization of the *imperceptible* adversarial examples found by the l_∞ Square Attack on ImageNet using ResNet-50 for $\epsilon_\infty = 4/255$. All the original images were correctly classified while the adversarial images are misclassified by the model. The perturbations are amplified for the visualization purpose

significant decrease of the loss. As the attack progresses over iterations, the size of the squares is reduced according to our piecewise-constant schedule leading to more refined perturbations since the algorithm accumulates a larger number of square-shaped updates.

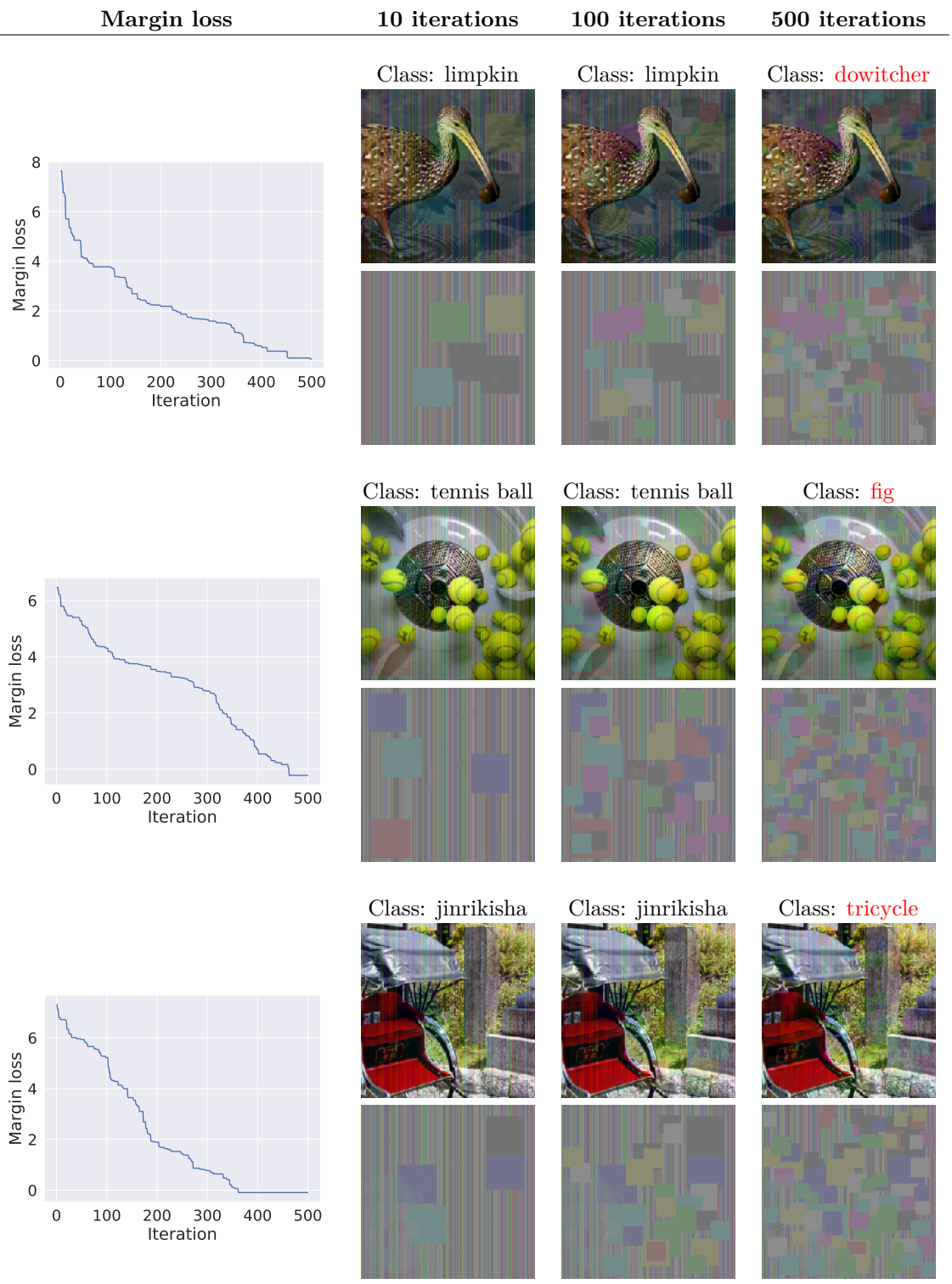


Figure 3.8: Visualization of adversarial examples for which the untargeted l_∞ Square Attack requires more queries. We visualize adversarial examples and perturbations after 10, 100 and 500 iterations of the attack. The experiment is done on ImageNet using ResNet-50 for $\epsilon_\infty = 0.05$. Note that a misclassification is achieved when the margin loss becomes negative

3.12.6 Breaking the Post-averaging Defense

We investigate whether the l_∞-robustness claims of Lin et al. (2019) hold (as reported in https://www.robust-ml.org/preprints/). Their defense method is a randomized averaging method similar in spirit to (Cohen et al., 2019). The difference is that (Lin et al., 2019) sample from the surfaces of several d-dimensional spheres instead of the Gaussian distribution, and they do not derive any robustness certificates, but rather measure robustness by the PGD attack. We use the hyperparameters specified in their code (K=15, R=6 on CIFAR-10 and K=15, R=30 on ImageNet). We show in Table 3.12 that the proposed defense can be broken by the l_∞-Square Attack, which is able to reduce the robust accuracy suggested by PGD from 88.4% to 15.8% on CIFAR-10 and from 76.1% to 0.4% on ImageNet (we set $p = 0.3$ for our attack). This again highlights that straightforward application of gradient-based white-box attacks may lead to inaccurate robustness estimation, and usage of the Square Attack can prevent false robustness claims.

Table 3.12: l_∞-robustness of the post-averaging randomized defense (Lin et al., 2019). The Square Attack shows that these models are not robust

ϵ_∞	**Dataset**	**Robust accuracy**		
		Clean	PGD	**Square**
8/255	CIFAR-10	92.6%	88.4%	**15.8%**
	ImageNet	77.3%	76.1%	**0.4%**

4 RobustBench: a Standardized Adversarial Robustness Benchmark

4.1 Preface

In this chapter, based on Croce et al. (2021), we focus on the problem of benchmarking adversarial robustness of machine learning models. We introduce `RobustBench`, a standardized benchmark for adversarial robustness, which aims to provide a reliable evaluation of the robustness of the considered models within a reasonable computational budget.

Summary As a research community, we are still lacking a systematic understanding of the progress on adversarial robustness which often makes it hard to identify the most promising ideas in training robust models. A key challenge in benchmarking robustness is that its evaluation is often error-prone leading to *robustness overestimation*. Our goal is to establish a *standardized benchmark* of adversarial robustness, which as accurately as possible reflects the robustness of the considered models within a reasonable computational budget. To this end, we start by considering the image classification task and introduce restrictions (possibly loosened in the future) on the allowed models. We evaluate adversarial robustness with *AutoAttack* (Croce and Hein, 2020b), an ensemble of white- and black-box attacks, which was recently shown in a large-scale study to improve almost all robustness evaluations compared to the original publications. To prevent overadaptation of new defenses to AutoAttack, we welcome external evaluations based on adaptive attacks (Tramèr et al., 2020), especially where AutoAttack flags a potential overestimation of robustness. Our leaderboard, hosted at `https://robustbench.github.io/`, contains evaluations of 120+ models and aims at reflecting the current state of the art in image classification on a set of well-defined tasks in ℓ_∞- and ℓ_2-threat models and on common corruptions, with possible extensions in the future. Additionally, we open-source the library `https://github.com/RobustBench/robustbench` that provides unified access to 80+ robust models to facilitate their downstream applications. Finally, based on the collected models, we analyze the impact of robustness on the performance on distribution shifts, calibration, out-of-distribution detection, fairness, privacy leakage, smoothness, and transferability.

Co-authors Francesco Croce, Vikash Sehwag, Edoardo Debenedetti, Nicolas Flammarion, Mung Chiang, Prateek Mittal, Matthias Hein.

Contributions Maksym Andriushchenko, Francesco Croce, and Vikash Sehwag came up with the project idea, implemented the `RobustBench` library (with the help from Edoardo Debenedetti), and jointly conducted experiments.

4.2 Introduction

Since the finding that state-of-the-art deep learning models are vulnerable to small input perturbations called *adversarial examples* (Szegedy et al., 2014), achieving adversarially robust models has become one of the most studied topics in the machine learning community. The main difficulty of robustness evaluation is that it is a computationally hard problem even for simple ℓ_p-bounded perturbations (Katz et al., 2017) and exact approaches (Tjeng et al., 2019a) do not scale to large enough models. There are already more than 3000 papers on this topic (Carlini, 2021), but it is often unclear which defenses against adversarial examples indeed improve robustness and which only make the typically used attacks overestimate the actual robustness. There is an important line of work on recommendations for how to perform adaptive attacks that are selected specifically for a particular defense (Athalye et al., 2018; Carlini et al., 2019a; Tramèr et al., 2020) which have in turn shown that several seemingly robust defenses fail to be robust. However, recently Tramèr et al. (2020) observe that although several recently published defenses have tried to perform adaptive evaluations, many of them could still be broken by new adaptive attacks. We observe that there are repeating patterns in many of these defenses that prevent standard attacks from succeeding. This motivates us to impose restrictions on the defenses we consider in our proposed benchmark, `RobustBench`, which aims at *standardized* adversarial robustness evaluation. Specifically, we rule out (1) classifiers which have zero gradients with respect to the input (Buckman et al., 2018; Guo et al., 2018), (2) randomized classifiers (Yang et al., 2019; Pang et al., 2020b), and (3) classifiers that use an optimization loop at inference time (Samangouei et al., 2018; Li et al., 2019c). Often, non-certified defenses that violate these three restrictions only make gradient-based attacks harder but do not substantially improve robustness (Carlini et al., 2019a). However, we will lift (some of) these constraints if a standardized reliable evaluation method for those defenses becomes available. We start from benchmarking robustness with respect to the ℓ_∞- and ℓ_2-threat models, since they are the most studied settings in the literature. We use the recent AutoAttack (Croce and Hein, 2020b) as our current standard evaluation which is an ensemble of diverse parameter-free attacks (white- and black-box) that has shown reliable performance over a large set of models that satisfy our restrictions. Moreover, we accept and encourage external evaluations, e.g. with adaptive attacks, to improve our standardized evaluation, especially for the leaderboard entries whose evaluation may be unreliable according to the *flag* that we propose. Additionally, we collect models robust against common image corruptions (Hendrycks and Dietterich, 2019) as these represent another important type of perturbations which should not change the

Rank	Method	Standard accuracy	AutoAttack robust accuracy	Best known robust accuracy	AA eval. potentially unreliable	Extra data	Architecture	Venue
1	Fixing Data Augmentation to Improve Adversarial Robustness *66.56% robust accuracy is due to the original evaluation (AutoAttack + MultiTargeted)*	92.23%	66.58%	66.56%	×	☑	WideResNet-70-16	arXiv, Mar 2021
2	Uncovering the Limits of Adversarial Training against Norm-Bounded Adversarial Examples *65.87% robust accuracy is due to the original evaluation (AutoAttack + MultiTargeted)*	91.10%	65.88%	65.87%	×	☑	WideResNet-70-16	arXiv, Oct 2020
3	Fixing Data Augmentation to Improve Adversarial Robustness *It uses additional 1M synthetic images in training. 64.58% robust accuracy is due to the original evaluation (AutoAttack + MultiTargeted)*	88.50%	64.64%	64.58%	×	×	WideResNet-106-16	arXiv, Mar 2021

Figure 4.1: The top-3 entries of our CIFAR-10 leaderboard hosted at https://robustbench.github.io/ for the ℓ_∞-perturbations of radius $\varepsilon_\infty = 8/255$.

decision of a classifier.

Contributions. We make following key contributions with our `RobustBench` benchmark:

- **Leaderboard** (https://robustbench.github.io/): a website with the leaderboard (see Fig. 4.1) based on *more than 120* evaluations where it is possible to track the progress and the current state of the art in adversarial robustness based on a standardized evaluation using AutoAttack *complemented* by (external) adaptive evaluations. The goal is to clearly identify the most successful ideas in training robust models to accelerate the progress in the field.

- **Model Zoo** (https://github.com/RobustBench/robustbench): a collection of the most robust models that are easy to use for any downstream applications. As an example, we expect that this will foster the development of better adversarial attacks by making it easier to perform evaluations on a large set of *more than 80* models.

- **Analysis**: based on the collected models from the Model Zoo, we provide an analysis of how robustness affects the performance on distribution shifts, calibration, out-of-distribution detection, fairness, privacy leakage, smoothness, and transferability. In particular, we find that robust models are significantly *underconfident* that leads to worse calibration, and that not all robust models have higher privacy leakage than standard models.

4.3 Background and related work

Adversarial perturbations. Let $\boldsymbol{x} \in \mathbb{R}^d$ be an input point and $y \in \{1, \dots, C\}$ be its correct label. For a classifier $f : \mathbb{R}^d \to \mathbb{R}^C$, we define a *successful adversarial perturbation*

with respect to the perturbation set $\Delta \subseteq \mathbb{R}^d$ as a vector $\boldsymbol{\delta} \in \mathbb{R}^d$ such that

$$\underset{c \in \{1,\dots,C\}}{\arg\max}\, f(\boldsymbol{x} + \boldsymbol{\delta})_c \neq y \quad \text{and} \quad \boldsymbol{\delta} \in \Delta, \tag{4.1}$$

where typically the perturbation set Δ is chosen such that *all* points in $x + \delta$ have y as their true label. This motivates a typical robustness measure called *robust accuracy*, which is the fraction of datapoints on which the classifier f predicts the correct class for all possible perturbations from the set Δ. Computing the exact robust accuracy is in general intractable and, when considering ℓ_p-balls as Δ, NP-hard even for single-layer neural networks (Katz et al., 2017; Weng et al., 2018). In practice, an *upper bound* on the robust accuracy is computed via some *adversarial attacks* which are mostly based on optimizing some differentiable loss (e.g., cross entropy) using local search algorithms like projected gradient descent (PGD) in order to find a successful adversarial perturbation. The tightness of the upper bound depends on the effectiveness of the attack: unsuitable techniques or suboptimal parameters (e.g., the step size and the number of iterations) can make the models appear more robust than they actually are (Engstrom et al., 2018; Mosbach et al., 2018), especially in the presence of phenomena like gradient obfuscation (Athalye et al., 2018). Certified methods (Wong and Kolter, 2018; Gowal et al., 2019a) instead provide *lower bounds* on robust accuracy which often underestimate robustness significantly, especially if the certification was not part of the training process. Thus, in our benchmark, we do not measure lower bounds and focus only on upper bounds which are typically much tighter (Tjeng et al., 2019a).

Threat models. We focus on the fully white-box setting, i.e. the model f is assumed to be fully known to the attacker. The threat model is defined by the set Δ of the allowed perturbations: the most widely studied ones are the ℓ_p-perturbations, i.e. $\Delta_p = \{\boldsymbol{\delta} \in \mathbb{R}^d, \|\boldsymbol{\delta}\|_p \leq \varepsilon\}$, particularly for $p = \infty$ (Szegedy et al., 2014; Goodfellow et al., 2015; Madry et al., 2018). We rely on thresholds ε established in the literature which are chosen such that the true label should stay the same for each in-distribution input within the perturbation set. We note that robustness towards small ℓ_p-perturbations is a necessary but not sufficient notion of robustness which has been criticized in the literature (Gilmer et al., 2018). It is an active area of research to develop threat models which are more aligned with the human perception such as spatial perturbations (Fawzi and Frossard, 2015; Engstrom et al., 2019c), Wasserstein-bounded perturbations (Wong et al., 2019; Hu et al., 2020), perturbations of the image colors (Laidlaw and Feizi, 2019) or ℓ_p-perturbations in the latent space of a neural network (Laidlaw et al., 2020; Wong and Kolter, 2020). However, despite the simplicity of the ℓ_p-perturbation model, it has numerous interesting applications that go beyond security considerations (Tramèr et al., 2019; Saadatpanah et al., 2020) and span transfer learning (Salman et al., 2020; Utrera et al., 2020), interpretability (Tsipras et al., 2019; Kaur et al., 2019; Engstrom et al., 2019b), generalization (Xie et al., 2020; Zhu et al., 2019; Bochkovskiy et al., 2020), robustness to unseen perturbations (Kang et al., 2019a; Xie et al., 2020; Laidlaw et al., 2020; Kireev et al., 2021), stabilization of GAN training (Zhong et al., 2020). Thus, improvements in ℓ_p-robustness have the potential to improve many of these downstream applications.

Additionally, we provide leaderboards for *common image corruptions* (Hendrycks and Dietterich, 2019) that try to mimic modifications of the input images which can occur naturally. Unlike ℓ_p adversarial perturbations, they are not imperceptible and evaluation on them is done in the average-case fashion, i.e. there is no attacker who aims at changing the classifier's decision. In this case, the robustness of a model is evaluated as classification accuracy on the corrupted images, averaged over corruption types and severities.

Related libraries and benchmarks. There are many libraries that focus primarily on implementations of popular adversarial attacks such as FoolBox (Rauber et al., 2017), Cleverhans (Papernot et al., 2018), AdverTorch (Ding et al., 2019), AdvBox (Goodman et al., 2020), ART (Nicolae et al., 2018), SecML (Melis et al., 2019), DeepRobust Li et al. (2020a). Some of them also provide implementations of several basic defenses, but they do not include up-to-date state-of-the-art models. The two challenges (Kurakin et al., 2018; Brendel et al., 2018b) hosted at NeurIPS 2017 and 2018 aimed at finding the most robust models for specific attacks, but they had a predefined deadline, so they could capture the best defenses only at the time of the competition. Ling et al. (2019) proposed DEEPSEC, a benchmark that tests many combinations of attacks and defenses, but suffers from a few shortcomings as suggested by Carlini (2019): (1) reporting average-case instead of worst-case performance over multiple attacks, (2) evaluating robustness in threat models different from the ones used for training, (3) using excessively large perturbations. Chen and Gu (2020) proposed a new *hard-label* black-box attack, RayS, and evaluated it on a range of models which led to a leaderboard (`https://github.com/uclaml/RayS`). Despite being a state-of-the-art hard-label black-box attack, the robust accuracy in the leaderboard given by RayS still tends to be overestimated even compared to the original evaluations.

Recently, Dong et al. (2020) have provided an evaluation of a few defenses (in particular, 3 for ℓ_∞- and 2 for ℓ_2-norm on CIFAR-10) against multiple commonly used attacks. However, they did not include some of the best performing defenses (Hendrycks et al., 2019; Carmon et al., 2019; Gowal et al., 2020; Rebuffi et al., 2021) and attacks (Gowal et al., 2019b; Croce and Hein, 2020a), and in a few cases, their evaluation suggests robustness higher than what was reported in the original papers. Moreover, they do not impose any restrictions on the models they accept to the benchmark. RobustML (`https://www.robust-ml.org/`) aims at collecting robustness claims for defenses together with external evaluations. Their format does not assume running any baseline attack, so it relies entirely on evaluations submitted by the community, which however do not occur often enough. Thus even though RobustML has been a valuable contribution to the community, now it does not provide a comprehensive overview of the recent state of the art in adversarial robustness.

Finally, it has become common practice to test new attacks wrt ℓ_∞ on the publicly available models from Madry et al. (2018) and Zhang et al. (2019c), since those represent widely accepted defenses which have stood many thorough evaluations. However, having only two models per dataset (MNIST and CIFAR-10) does not constitute a sufficiently large testbed, and, because of the repetitive evaluations, some attacks may already overfit to

those defenses.

What is different in RobustBench. Learning from these previous attempts, RobustBench presents a few different features compared to the aforementioned benchmarks: (1) a baseline worst-case evaluation with an ensemble of *strong*, *standardized* attacks (Croce and Hein, 2020b) which includes both white- and black-box attacks, unlike RobustML which is *solely* based on adaptive evaluations, integrated by external evaluations, (2) we add a *flag* in AutoAttack raised when the evaluation might be unreliable, in which case we do additional adaptive evaluations ourselves and encourage the community to contribute, (3) clearly defined threat models that correspond to the ones used during training of submitted models, (4) evaluation of not only standard defenses (Madry et al., 2018; Zhang et al., 2019c) but also of more recent improvements such as (Carmon et al., 2019; Gowal et al., 2020; Rebuffi et al., 2021). Moreover, RobustBench is designed as an *open-ended* benchmark that keeps an up-to-date leaderboard, and we welcome contributions of new defenses and evaluations using adaptive attacks. Finally, we open source the Model Zoo for convenient access to the 80+ most robust models from the literature which can be used for downstream tasks and facilitate the development of new standardized attacks.

4.4 Description of RobustBench

We start by providing a detailed layout of our proposed leaderboards for ℓ_∞, ℓ_2, and common corruption threat models. Next, we present the Model Zoo, which provides unified access to most networks from our leaderboards.

4.4.1 Leaderboard

Restrictions. We argue that accurate benchmarking adversarial robustness in a standardized way requires some restrictions on the type of considered models. The goal of these restrictions is to prevent submissions of defenses that cause some standard attacks to fail without truly improving robustness. Specifically, we consider only classifiers $f : \mathbb{R}^d \to \mathbb{R}^C$ that

- have in general *non-zero gradients* with respect to the inputs. Models with zero gradients, e.g., that rely on quantization of inputs (Buckman et al., 2018; Guo et al., 2018), make gradient-based methods ineffective thus requiring zeroth-order attacks, which do not perform as well as gradient-based attacks. Alternatively, specific adaptive evaluations, e.g. with Backward Pass Differentiable Approximation (Athalye et al., 2018), can be used which, however, can hardly be standardized. Moreover, we are not aware of existing defenses solely based on having zero gradients for large parts of the input space which would achieve competitive robustness.
- have a *fully deterministic forward pass*. To evaluate defenses with stochastic components, it is a common practice to combine standard gradient-based attacks with

Expectation over Transformations (Athalye et al., 2018). While often effective it might be not sufficient, as shown by Tramèr et al. (2020). Moreover, the classification decision of randomized models may vary over different runs for the same input, hence even the definition of robust accuracy differs from that of deterministic networks. We note that randomization *can* be useful for improving robustness and deriving robustness certificates (Lecuyer et al., 2019; Cohen et al., 2019), but it also introduces variance in the gradient estimators (both white- and black-box) making standard attacks much less effective.

- do not have an *optimization loop* in the forward pass. This makes backpropagation through it very difficult or extremely expensive. Usually, such defenses (Samangouei et al., 2018; Li et al., 2019c) need to be evaluated adaptively with attacks that rely on a combination of hand-crafted losses.

Some of these restrictions were also discussed by Brown et al. (2018) for the warm-up phase of their challenge. We refer the reader to Appendix E therein for an illustrative example of a trivial defense that bypasses gradient-based and some of the black-box attacks they consider. We believe that such constraints are necessary at the moment since they allow an accurate standardized evaluation which makes the leaderboard meaningful and sustainable. In fact, for defenses not fulfilling the restrictions there is no standard evaluation which is shown to generalize and perform well across techniques, thus one has to resort to time-consuming adaptive attacks specifically tailored for each case. In the design of our benchmark, we thought that it is more important that the robustness evaluation is reliable, rather than being open to all possible defenses with the risk that the robustness is drastically overestimated. As this can lead to a potential bias in our leaderboard, we will lift the restrictions if reliable standardized evaluation methods for these modalities become available in the literature.

Overall setup. We set up leaderboards for the ℓ_∞, ℓ_2 and common corruption threat models on CIFAR-10, CIFAR-100 (Krizhevsky and Hinton, 2009), and ImageNet Deng et al. (2009) datasets (see Table 4.1 for details). We use the fixed budgets of $\varepsilon_\infty = 8/255$ and $\varepsilon_2 = 0.5$ for the ℓ_∞ and ℓ_2 leaderboards for CIFAR-10 and CIFAR-100. For ImageNet, we use $\varepsilon_\infty = 4/255$ and in App. 4.10, we discuss how we handle that different models use different image resolutions for ImageNet. Most of the models shown in the leaderboards are taken from papers published at top-tier machine learning and computer vision conferences as shown in Fig. 4.2 (left). For each entry we report the reference to the original paper, standard and robust accuracy under the specific threat model (see the next paragraph for details), network architecture, venue where the paper appeared and possibly notes regarding the model. We also highlight when extra data (often, the dataset introduced by Carmon et al. (2019)) is used since it gives a clear advantage for both clean and robust accuracy. If any other attack achieves lower robust accuracy than AutoAttack then we also report it. Moreover, the leaderboard allows to search the entries by their metadata (such as title, architecture, venue) which can be useful to compare different methods that use the same architecture or to search for papers published at some conference.

Figure 4.2: Visualization of the robustness and accuracy of 54 CIFAR-10 models from the `RobustBench` ℓ_∞-leaderboard. Robustness is evaluated using ℓ_∞-perturbations with $\varepsilon_\infty = 8/255$.

Evaluation of defenses. The evaluation of robust accuracy on common corruptions (Hendrycks and Dietterich, 2019) involves simply computing the average accuracy on corrupted images over different corruption types and severity levels.[1] To evaluate robustness of ℓ_∞ and ℓ_2 defenses, we currently use AutoAttack (Croce and Hein, 2020b). It is an ensemble of four attacks that are run sequentially: a variation of PGD attack with automatically adjusted step sizes, with (1) the cross entropy loss and (2) the difference of logits ratio loss, which is a rescaling-invariant margin-based loss function, (3) the targeted version of the FAB attack (Croce and Hein, 2020a), which minimizes the ℓ_p-norm of the perturbations, and (4) the black-box Square Attack (Andriushchenko et al., 2020). Each subsequent attack is run on the points for which an adversarial example has not been found by the preceding attacks. We choose AutoAttack as it includes both black-box and white-box attacks, does not require hyperparameter tuning (in particular, the step size), and consistently improves the results reported in the original papers for almost all the models (see Fig. 4.2 (middle)). If in the future some new standardized and parameter-free attack is shown to consistently outperform AutoAttack on a wide set of models given a similar computational cost, we will adopt it as standard evaluation. In order to verify the reproducibility of the results, we perform the standardized evaluation independently of the authors of the submitted models. We encourage evaluations of the models in the leaderboard with adaptive or external attacks to reflect the best available upper bound on the true robust accuracy (see a pre-formatted issue template in our repository[2]), in particular in the case where AutoAttack flags that it might not be reliable (see paragraph below). For example, Gowal et al. (2020) and Rebuffi et al. (2021) evaluate their models with a hybrid of AutoAttack and MultiTargeted attack (Gowal et al., 2019b), that in some cases reports slightly lower robust accuracy than AutoAttack alone. We reflect the additional evaluations in our leaderboard by reporting in a separate column the robust accuracy for the worst case of AutoAttack and all other evaluations. Below we show an example of how one can use our library to easily benchmark a model (either external one or taken from the Model Zoo):

[1] A breakdown over corruptions and severities is also available, e.g. for CIFAR-10 models see: https://github.com/RobustBench/robustbench/blob/master/model_info/cifar10/corruptions/unaggregated_results.csv

[2] https://github.com/RobustBench/robustbench/issues/new/choose

```
from robustbench.eval import benchmark
clean_acc, robust_acc = benchmark(model, dataset='cifar10', threat_model='Linf')
```

Moreover, in Appendix 4.11 we also show the variability of the robust accuracy given by AutoAttack over random seeds and report its runtime for a few models from different threat models.

Identifying potential need for adaptive attacks. Although AutoAttack provides an accurate estimation of robustness for most models that satisfy the restrictions mentioned above, there might still be corner cases when AutoAttack overestimates robustness of a model that satisfies the restrictions. Carlini et al. (2019a) suggest that one indicator of possible overestimation of robustness is when black-box attacks are more effective than white-box ones. We noticed that this is the case for the model from Xiao et al. (2020) where the black-box Square Attack (Andriushchenko et al., 2020) improves by *more than* 10% the robust accuracy given by the previous white-box attacks in AutoAttack. We run a simple adaptive attack: Square Attack with multiple random restarts (30 instead of 1) decreases the robust accuracy from the 18.50% of AutoAttack to 7.40%. We note that AutoAttack did not fail completely for this model and correctly revealed a lower level of robustness than reported (52.4%), although the exact robust accuracy was overestimated. Based on this case, we integrate a *flag* in AutoAttack: a warning is output whenever Square Attack reduces of more than 0.2% the robust accuracy compared to the white-box gradient-based attacks in AutoAttack. In this case, AutoAttack evaluation might be not fully reliable and adaptive attacks might be necessary, so we flag the corresponding entries in the leaderboard (currently, only the model of Xiao et al. (2020)). Moreover, for the sake of convenience, we also integrate in AutoAttack flags that automatically inform the user if the restrictions are violated.[3]

Adding new defenses. We believe that the leaderboard is only useful if it reflects the latest advances in the field, so it needs to be constantly updated with new defenses. We intend to include evaluations of new techniques and we welcome contributions from the community which help to keep the benchmark up-to-date. We require new entries to (1) satisfy the three restrictions stated above, (2) to be accompanied by a publicly available paper (e.g., an arXiv preprint) describing the technique used to achieve the reported results, and (3) share the model checkpoints (not necessarily publicly). We also allow *temporarily* adding entries without providing checkpoints given that the authors evaluate their models with AutoAttack. However, we will mark such evaluations as *unverified*, and to encourage reproducibility, we reserve the right to remove an entry later on if the corresponding model checkpoint is not provided. It is possible to add a new defense to the leaderboard and (optionally) the Model Zoo by opening an issue with a predefined template in our repository https://github.com/RobustBench/robustbench, where more details about new additions can be found.

[3]See https://github.com/fra31/auto-attack/blob/master/flags_doc.md for details

Table 4.1: The total number of models in the Model Zoo and leaderboards per dataset and threat model.

	CIFAR-10		CIFAR-100		ImageNet	
Threat model	Model Zoo	Leaderboard	Model Zoo	Leaderboard	Model Zoo	Leaderboard
ℓ_∞	39	63	14	14	5	6
ℓ_2	17	18	-	-	-	-
Common corruptions	7	15	2	4	5	7

4.4.2 Model Zoo

We collect the checkpoints of many networks from the leaderboard in a single repository hosted at https://github.com/RobustBench/robustbench after obtaining the permission of the authors (see Appendix 4.8 for the information on the licenses). The goal of this repository, the Model Zoo, is to make the usage of robust models as simple as possible to facilitate various downstream applications and analyses of general trends in the field. In fact, even when the checkpoints of the proposed method are made available by the authors, it is often time-consuming and not straightforward to integrate them in the same framework because of many factors such as small variations in the architectures, custom input normalizations, etc. For simplicity of implementation, at the moment we include only models implemented in PyTorch (Paszke et al., 2017). Below we illustrate how a model can be automatically downloaded and loaded via its identifier and threat model within two lines of code:

```
from robustbench.utils import load_model
model = load_model(model_name='Ding2020MMA', dataset='cifar10', threat_model='L2')
```

At the moment, all models (see Table 4.1 and Appendix 4.13 for details) are variations of ResNet (He et al., 2016a) and WideResNet architectures (Zagoruyko and Komodakis, 2016) of different depth and width. However, we note that the benchmark and Model Zoo are not restricted only to residual or convolutional networks, and we are ready to add any other architecture. We include the most robust models, e.g. those from Rebuffi et al. (2021), but there are also defenses which pursue additional goals alongside adversarial robustness at the fixed threshold we use: e.g., Sehwag et al. (2020b) consider networks which are robust and compact, Wong et al. (2020) focus on computationally efficient adversarial training, Ding et al. (2020) aim at input-adaptive robustness as opposed to robustness within a single ℓ_p-radius. All these factors have to be taken into account when comparing different techniques, as they have a strong influence on the final performance. Thus, we highlight these factors in the footnotes below each paper's title.

A testbed for new attacks. Another important use case of the Model Zoo is to simplify comparisons between different adversarial attacks on a wide range of models. First, the leaderboard already serves as a strong baseline for new attacks. Second, as mentioned above, new attacks are often evaluated on the models from Madry et al. (2018) and Zhang et al. (2019c), but this may not provide a representative picture of their effectiveness. For

example, currently the difference in robust accuracy between the first and second-best attacks in the CIFAR-10 leaderboard of Madry et al. (2018) is only 0.03%, and between the second and third is 0.04%. Thus, we believe that a more thorough comparison should involve multiple models to prevent overfitting of the attack to one or two standard robust defenses.

4.5 Analysis

With unified access to multiple models from the Model Zoo, one can easily compute various performance metrics to see general trends. We analyze various aspects of robust classifiers, mostly for ℓ_∞-robust models on CIFAR-10. Results for other threat models and datasets can be found in App. 4.12.

Progress on adversarial defenses. In Fig. 4.2, we plot a breakdown over conferences, the amount of robustness overestimation reported in the original papers, and we also visualize the robustness-accuracy trade-off for the ℓ_∞-models from the Model Zoo. First, we observe that for multiple *published* defenses, the reported robust accuracy is highly overestimated. We also find that the use of extra data is able to alleviate the robustness-accuracy trade-off as suggested in previous works (Raghunathan et al., 2020). However, so far all models with high robustness to perturbations of ℓ_∞-norm up to $\varepsilon = 8/255$ still suffer from noticeable degradation in clean accuracy compared to standardly trained models. Finally, it is interesting to note that the best entries of the ℓ_p-leaderboards are still variants of PGD adversarial training (Madry et al., 2018; Zhang et al., 2019c) but with various enhancements (extra data, early stopping, weight averaging).

Performance across various distribution shifts. We test the performance of the models from the Model Zoo on different distribution shifts ranging from common image corruptions (CIFAR-10-C, Hendrycks and Dietterich (2019)) to dataset resampling bias (CIFAR-10.1, Recht et al. (2019)) and image source shift (CINIC-10, Darlow et al. (2018)). For each of these datasets, we measure standard accuracy, and Fig. 4.3 shows that improvement in robust accuracy (which often comes with an improvement in standard accuracy) on CIFAR-10 correlates with an improvement in standard accuracy across distributional shifts. On CIFAR-10-C, robust models (particularly with respect to the ℓ_2-norm) tend to give a significant improvement which agrees with the findings in Ford et al. (2019). Concurrently with our work, Taori et al. (2020a) study the robustness to different distribution shifts of many models trained on ImageNet, including some ℓ_p-robust models. Our conclusions qualitatively agree with theirs, and we hope that our collected set of models will help to provide a more complete picture. Moreover, we measure robust accuracy, in the same threat model used on CIFAR-10, using AutoAttack (Croce and Hein, 2020b) (see Fig. 4.10 in Appendix 4.12), in order to see how ℓ_p adversarial robustness generalizes across the datasets representing different distributions shifts, and observe a clear positive correlation between robust accuracy on CIFAR-10 and its variations.

Calibration. A classifier is *calibrated* if its predicted probabilities correctly reflect the

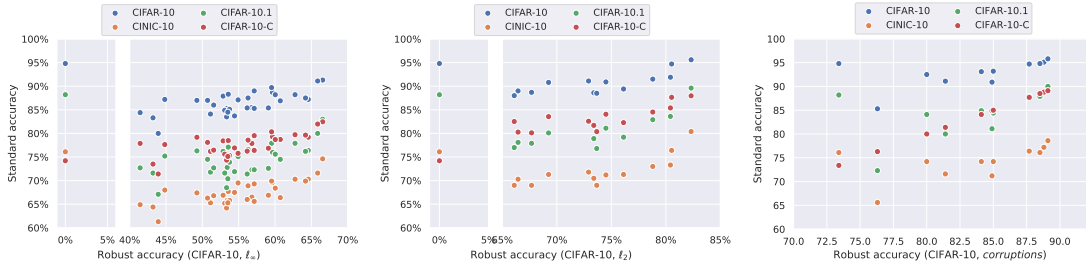


Figure 4.3: Standard accuracy of classifiers trained against ℓ_∞ (**left**), ℓ_2 (**middle**), and common corruption (**right**) threat model respectively, from our Model Zoo on various distribution shifts.

actual accuracy (Guo et al., 2017). In the context of adversarial training, calibration was considered in Hendrycks et al. (2020) who focus on improving accuracy on common corruptions and in Augustin et al. (2020) who focus mostly on preventing overconfident predictions on out-of-distribution inputs. We instead focus on *in-distribution* calibration, and in Fig. 4.4 plot the expected calibration error (ECE) without and with temperature rescaling (Guo et al., 2019b) to minimize the ECE (which is a simple but effective post-hoc calibration method, see Appendix 4.12 for details) together with the optimal temperature for a large set of ℓ_∞ models. We observe that most of the ℓ_∞ robust models are significantly *underconfident* since the optimal calibration temperature is less than one for most models. The only two models in Fig. 4.4 which are *overconfident* are the standard model and the model of Ding et al. (2020) that aims to maximize the margin. We see that temperature rescaling is even more important for robust models since without any rescaling the ECE is as high as 70% for the model of Pang et al. (2020c) (and 21% on average) compared to 4% for the standard model. Temperature rescaling significantly reduces the ECE gap between robust and standard models but it does not fix the problem completely which suggests that it is worth incorporating calibration techniques also during training of robust models. For ℓ_2 robust models, the models can be on the contrary *more calibrated* by default, although the improvement vanishes if temperature rescaling is applied (see Appendix 4.12).

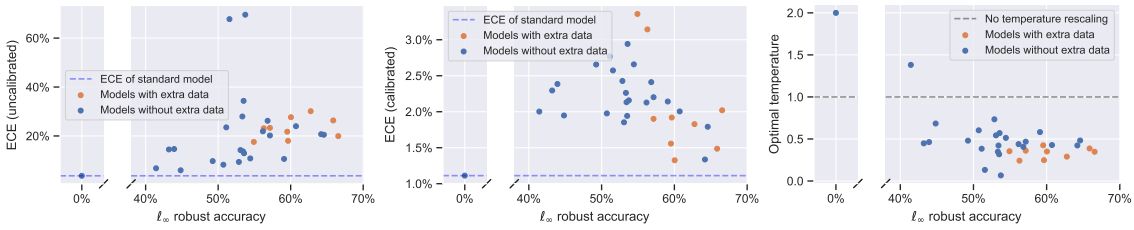


Figure 4.4: Expected calibration error (ECE) before (**left**) and after (**middle**) temperature rescaling, and the optimal rescaling temperature (**right**) for the ℓ_∞-robust models.

Out-of-distribution detection. Ideally, a classifier should exhibit high uncertainty in its predictions when evaluated on *out-of-distribution* (OOD) inputs. One of the most straightforward ways to extract this uncertainty information is to use some threshold on the predicted confidence where OOD inputs are expected to have low confidence from the model (Hendrycks and Gimpel, 2017). An emerging line of research aims at devel-

oping OOD detection methods in conjunction with adversarial robustness (Hein et al., 2019; Sehwag et al., 2019; Augustin et al., 2020). In particular, Song et al. (2020) demonstrated that adversarial training (Madry et al., 2018) leads to degradation in the robustness against OOD data. We further test this observation on all ℓ_∞-models trained on CIFAR-10 from the Model Zoo on three OOD datasets: CIFAR-100 (Krizhevsky and Hinton, 2009), SVHN (Netzer et al., 2011), and Describable Textures Dataset (Cimpoi et al., 2014). We use the area under the ROC curve (AUROC) to measure the success in the detection of OOD data, and show the results in Fig. 4.5. With ℓ_∞ robust models, we find that compared to standard training, various robust training methods indeed lead to degradation of the OOD detection quality. While extra data in standard training can improve robustness against OOD inputs, it fails to provide similar improvements with robust training. We further find that ℓ_2 robust models have in general comparable OOD detection performance to standard models (see Fig. 4.12 in Appendix), while the model of Augustin et al. (2020) achieves even better performance since their approach explicitly optimizes both robust accuracy and worst-case OOD detection performance.

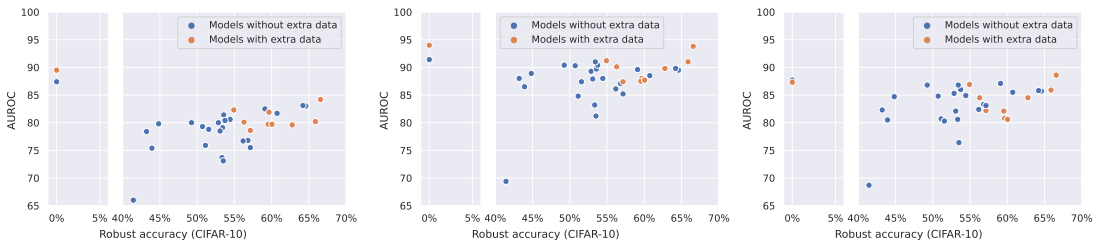


Figure 4.5: Visualization of the OOD detection quality (higher AUROC is better) for the ℓ_∞-robust models trained on CIFAR-10 on three OOD datasets: CIFAR-100 (**left**), SVHN (**middle**), Describable Textures (**right**). We detect OOD inputs based on the maximum predicted confidence (Hendrycks and Gimpel, 2017).

Fairness in robustness. Recent works (Benz et al., 2020; Xu et al., 2020) have noticed that robust training Madry et al. (2018); Zhang et al. (2019c) can lead to models whose performance varies significantly across subgroups, e.g. defined by classes. We will refer to this performance difference as *fairness*, and here we study the influence of robust training methods on fairness. In Fig. 4.6 we show the breakdown of standard and robust accuracy for the ℓ_∞ robust models, where one can see how the achieved robustness largely varies over classes. While in general the classwise standard and robust accuracy correlate well, the class "deer" in ℓ_∞-threat model suffers a significant degradation, unlike what happens for ℓ_2 (see Appendix 4.12), which might indicate that the features of such class are particularly sensitive to ℓ_∞-bounded attacks. Moreover, we measure fairness with the relative standard deviation (RSD), defined as the standard deviation divided over the average, of robust accuracy over classes for which lower values mean more uniform distribution and higher robustness. We observe that better robust accuracy generally leads to lower RSD values which implies that the disparity among classes is reduced. (with a strong linear trend): improving the robustness of the models has then the effect of reducing the disparities among classes. However, some training techniques like MART Wang et al. (2020) can

noticeably increase the RSD and thus *increase the disparity* compared to other methods which achieve similar robustness (around 57%).

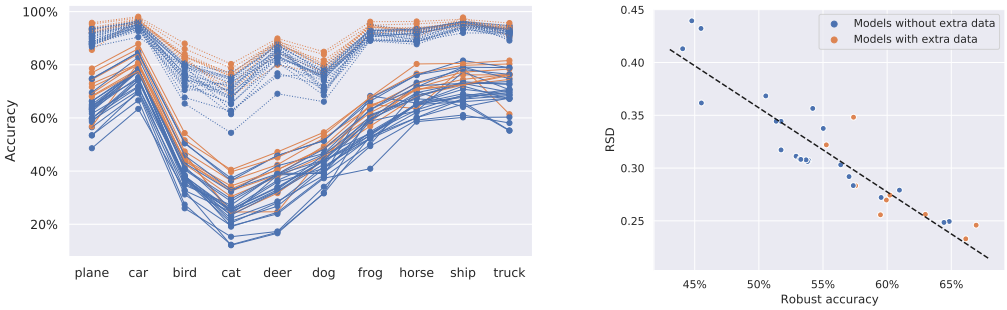


Figure 4.6: Fairness of ℓ_∞-robust models. **Left:** classwise standard (dotted lines) and robust (solid) accuracy. **Right:** relative standard deviation (RSD) of robust accuracy over classes vs its average.

Privacy leakage. Deep neural networks are prone to memorizing training data Shokri et al. (2017); Carlini et al. (2019b). Recent work has highlighted that robust training exacerbates this problem Song et al. (2019). We benchmark privacy leakage of training data across robust networks (Fig. 4.7). We calculate membership inference accuracy using output confidence of adversarial images from the training and test sets (see Appendix 4.12 for more details). It measures how accurately we can infer whether a sample was present in the training dataset. Our analysis reveals mixed trends. First, our results show that not all robust models have a significantly higher privacy leakage than a standard model. We find that the inference accuracy across robust models has a large variation, where some models even have lower privacy leakage than a standard model, and there is no strong correlation with robust accuracy. In contrast, it is largely determined by the generalization gap, as using the classifier confidence does not lead to a much higher inference accuracy than the baseline determined by the generalization gap (as shown in Fig. 4.7 (right)). Thus one can expect lower privacy leakage in robust networks as previous work explicitly aimed to reduce the generalization gap in robust training e.g. via early stopping Rice et al. (2020); Zhang et al. (2019c); Gowal et al. (2020).

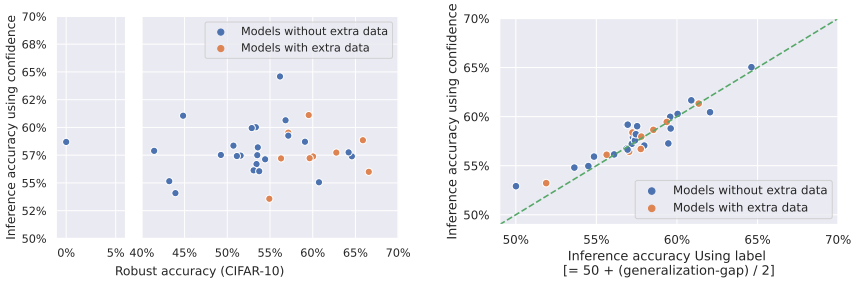


Figure 4.7: Privacy leakage of ℓ_∞-robust models. We measure privacy leakage of training data in robust networks and compare it with robust accuracy (**left**) and generalization gap (**right**).

Extra experiments. In Appendix 4.12, we show extra experiments related to the points analyzed above and describe some of the implementation details. Also, we study how

adversarial perturbations transfer between different models. We find that adversarial examples strongly transfer from robust to robust, non-robust to robust, and non-robust to non-robust networks. However, we observe poor transferability of adversarial examples from robust to non-robust networks. Finally, since prior works Hein and Andriushchenko (2017a); Yang et al. (2020) connected higher smoothness with better robustness, we analyze the smoothness of the models both at intermediate and output layers. This confirms that, for a fixed architecture, standard training yields classifiers that are significantly less smooth than robust ones. This study of properties of networks illustrates another useful aspect of our Model Zoo.

4.6 Outlook

Conclusions. We believe that a *standardized* benchmark with clearly defined threat models, restrictions on submitted models, and tight upper bounds on robust accuracy is useful to show which ideas in training robust models are the most successful. While AutoAttack is for most models very reliable and accurate and allows a standardized comparison, we ensure by flagging potentially unreliable evaluations and doing additional adaptive attacks that the benchmark reflects the best possible robustness assessment with limited resources as the exact robustness evaluation is computationally infeasible. We remark that recent works have already referred to our leaderboards Koh et al. (2020); Yu et al. (2021); Maho et al. (2021); Tao et al. (2021); Xu et al. (2021), in particular as reflecting the current state of the art Rebuffi et al. (2021); Li et al. (2021); Pang et al. (2021b), and used the networks of our Model Zoo to test new adversarial attacks Melis et al. (2019); Rony et al. (2020); Faghri et al. (2021); Schwinn et al. (2021), evaluate test-time defenses Wang et al. (2021) or perceptual distances derived from them Ju (2021), explore further properties of robust models Stutz et al. (2021); Zhang and Evans (2021). Additionally, we have shown that unified access to a *large* and *up-to-date* set of robust models can be useful to analyze multiple aspects related to robustness. First, one can easily analyze the progress of adversarial defenses over time including the amount of robustness overestimation and the robustness-accuracy tradeoff. Second, one can conveniently study the impact of robustness on other performance metrics such as accuracy under distribution shifts, calibration, out-of-distribution detection, fairness, privacy leakage, smoothness, and transferability. Overall, we think that the community has to develop a better understanding of how different types of robustness affect other aspects of the model performance and `RobustBench` can help to achieve this goal. Finally, we note that a good performance on our benchmark does not guarantee the safety of the benchmarked model in a real-world deployment since ℓ_p- and corruption robustness may not be sufficiently representative of all realistic threat models.

Future plans. Our intention in the future is to keep the current leaderboards up-to-date (see the maintenance plan in Appendix 4.9) and add new leaderboards for other datasets and other threat models which become widely accepted in the community. We see as potential candidates (1) sparse perturbations, e.g. bounded by ℓ_0, ℓ_1-norm or adversar-

ial patches (Brown et al., 2017; Croce and Hein, 2019; Modas et al., 2019; Croce et al., 2020), (2) multiple ℓ_p-norm perturbations (Tramèr and Boneh, 2019; Maini et al., 2020), (3) adversarially optimized common corruptions (Kang et al., 2019a,b), (4) a broad set of perturbations unseen during training (Laidlaw et al., 2020). Another possible direction is the development of a standardized evaluation of recent defenses based on some form of test-time adaptation (Shi et al., 2021; Wang et al., 2021), which do not fulfill the third restriction (no optimization loop). Finally, although the benchmark currently focuses on image classification, we think that its structure and principles should apply to other tasks (e.g., image segmentation (Xie et al., 2017), image retrieval (Tolias et al., 2019)) and domains (e.g., natural language processing (Alzantot et al., 2018), malware detection (Grosse et al., 2017)) where adversarial robustness can be of interest. Since this direction requires more domain-specific expertise, we welcome contributions from different communities to expand RobustBench.

Appendix

4.7 Broader impact

We note that the restrictions we impose on the defenses allowed in our benchmark could lead to a potential bias of the community which discourages research in certain directions. It is certainly not our goal to discourage research in directions which violate restrictions of the benchmark. However, without these restrictions a reliable evaluation of adversarial robustness is not feasible and a reliable evaluation of adversarial robustness in order to identify true advances in the field is key for further progress. Thus we think that these restrictions are unavoidable for a benchmark but we are working on relaxing the restrictions as much as possible.

Additionally, in motivating higher robustness against adversarial examples, our work may leave an unwanted side effect on tasks where adversarial attacks can actually be used for beneficial purposes Kulynych et al. (2020); Shan et al. (2020); Rahman et al. (2020). However, this is true for any paper that aims at improving adversarial robustness (either directly or indirectly via, e.g., a standardized benchmark).

On the positive side, in our work, we do not only perform a standardized benchmarking of adversarial robustness but also analyze multiple other properties of robust models such as calibration, privacy leakage, fairness, etc. In our opinion, such analyses are important since they allow us to assess the broader impact of improving robustness on other crucial performance metrics of neural networks.

Finally, we note that a good performance on our benchmark does not guarantee the safety of the benchmarked model in a real-world deployment which is likely to require more domain-specific threat models. ℓ_p-bounded adversarial attacks can be a realistic threat model in applications where it is possible to input an image directly in a digital format (Tramèr et al., 2019; Saadatpanah et al., 2020). However, attacks in-the-wild (Kurakin et al., 2017; Evtimov et al., 2018) are usually much more involved and differ considerably from the presented simple ℓ_p-perturbations. Moreover, the common corruptions we used for evaluation from Hendrycks and Dietterich (2019) are artificially generated, and thus may differ from the corruptions encountered in the real world. Taking this into account, we suggest to always think critically about the robustness requirements that are necessary for a particular application at hand.

4.8 Licenses

The code used for benchmarking is released under MIT license. The code of AutoAttack (Croce and Hein, 2020b) that our benchmark relies on has been released under the MIT license as well. The classifiers in the Model Zoo are added according to the per-

mission given by the authors with the license they choose: most of the models have MIT license, other have more restrictive ones such as Attribution-NonCommercial-ShareAlike 4.0 International, Apache License 2.0, BSD 3-Clause License. The details can be found at https://github.com/RobustBench/robustbench/blob/master/LICENSE. The CIFAR-10 and CIFAR-100 datasets (Krizhevsky and Hinton, 2009) are obtained via the PyTorch loaders (Paszke et al., 2017), while CIFAR-10-C and CIFAR-100-C Hendrycks and Dietterich (2019), with the common corruptions, are downloaded from the official release (see https://zenodo.org/record/2535967#.YLYf9agzaUk and https://zenodo.org/record/3555552#.YLYeJagzaUk). The validation set of ImageNet is not hosted or downloaded by our provided evaluation code, but it needs to be downloaded in advance directly by the user.

4.9 Maintenance plan

Here we discuss the main aspects of maintaining RobustBench and the costs associated with it:

- **Hosting the website** (https://robustbench.github.io/): we host our leaderboard using GitHub pages[4] which is a free service.
- **Hosting the library** (https://github.com/RobustBench/robustbench): the code of our library is hosted on GitHub[5] which offers the basic features that we need to maintain the library for free.
- **Hosting the models**: to ensure the availability of the models from the Model Zoo, we host them in our own cloud storage on Google Drive[6]. At the moment, they take around 24 GB of space which fits into the 100 GB storage plan that costs 2 USD per month.
- **Running evaluations**: we run all evaluations on the GPU servers that are available to our research groups which incurs no extra costs.

Moreover, as we mention in the outlook (Sec. 4.6), we also plan to expand the benchmark to new datasets and threat models which can slightly increase the required maintenance costs since we may need to upgrade the storage plan. We also expect the benchmark to be community-driven and to encourage this we have provided instructions[7] on how to submit new entries to the leaderboard and to the Model Zoo.

[4] https://pages.github.com/
[5] https://github.com/
[6] https://www.google.com/drive/
[7] https://github.com/RobustBench/robustbench#adding-a-new-model

4.10 Details of the ImageNet leaderboards

Extending the benchmark to ImageNet presents some challenges compared to CIFAR-10 and CIFAR-100. First, the ImageNet validation set (usually used as the test set) contains *50'000* images which makes it infeasible to run expensive evaluations on it. Thus, we define a fixed subset (5'000 randomly sampled images in our case) for faster evaluation, whose image IDs we make available in the Model Zoo. Second, it is not obvious how to handle the fact that different models may use different preprocessing techniques (e.g., different resolution, cropping, etc) which makes the search space for an attack *not fully comparable* across defenses. For this, we decide to allow models with different preprocessing steps and input resolution, considering them yet another design choice similarly to the choice of the network architecture which also has a large influence on the final results. Since in the ℓ_∞-threat model the constraints are componentwise independent, we use the same threshold $\varepsilon_\infty = 4/255$ for every classifier, regardless of the input dimensionality which is used after preprocessing.

4.11 Reproducibility and runtime

Here we discuss the main aspects of the reproducibility of the benchmark.

First of all, the code to run the benchmark on a given model is available in our repository, and an example of how to run it is given in the README file. The installation instructions are also provided in the README file and the requirements will be installed automatically.

To satisfy other points from the reproducibility checklist[8] which are applicable to our benchmark, we also discuss next the variability of the robust accuracy over random seeds and the average runtime of the benchmark. Evaluation of the accuracy on common corruptions (Hendrycks and Dietterich, 2019) is deterministic if we do not take into account non-deterministic operations on computational accelerators such as GPUs[9] which, however, do not affect the resulting accuracy. On the other hand, robustness evaluation using AutoAttack has an element of randomness since it relies on random initialization of the starting points and also on the randomness in the update of the Square Attack (Andriushchenko et al., 2020). To show the effect of randomness on the robust accuracy given by AutoAttack, we repeat evaluation over four random seeds on four models available in the Model Zoo from different threat models covering all datasets considered. In Table 4.2, we report the average robust accuracy with its standard deviation and observe that different seeds lead to very similar results. Moreover, we indicate the runtime of each evaluation, which is largely influenced by the size of the model, the computing infrastructure (every run uses a single Tesla V100 GPU), and the dataset. Moreover, less robust models require less time for evaluation which is due to the fact that AutoAttack does not further attack a point if an adversarial example is already found by some preceding attack

[8] https://www.cs.mcgill.ca/~jpineau/ReproducibilityChecklist.pdf
[9] https://pytorch.org/docs/stable/notes/randomness.html

in the ensemble.

Additionally, as mentioned above, for ImageNet we have randomly sampled and fixed 5000 images from the validation set. We provide the IDs of those images and code to load them in our repository. Note that we use the same set of images for ℓ_p-robustness and for common corruptions, in which case for every point 15 types of corruptions at 5 severity levels are applied, consistently with the other datasets.

Finally, when we extended the benchmark to ImageNet, we noticed that different versions of `PyTorch` and `torchvision` may lead to small differences in the standard accuracy (up to 0.16% on 5000 points for the same model). We suspect this is due to minor variations in the implementation of the preprocessing functions (such as resizing). Thus, we fix in the requirements `torch==1.7.1` and `torchvision==0.8.2` to ensure reproducibility. Note that the overall *ranking* and level of robustness of the defenses should not be influenced by using different versions of these libraries. We have not noticed similar issues for the other datasets.

Table 4.2: Statistics about the standardized evaluation with AutoAttack when repeated for four random seeds. We can see that the robust accuracy has very small fluctuations. We also report the runtime for the different models which is much smaller for less robust models.

Dataset	Norm	Paper	Architecture	Clean acc.	Robust acc.	Time
CIFAR-10	ℓ_∞	Gowal et al. (2020)	WRN-28-10	89.48%	62.82% ± 0.016	11.8 h
CIFAR-10	ℓ_2	Rebuffi et al. (2021)	WRN-28-10	91.79%	78.80% ± 0.000	15.1 h
CIFAR-100	ℓ_∞	Wu et al. (2020b)	WRN-34-10	60.38%	28.84% ± 0.018	6.6 h
ImageNet	ℓ_∞	Salman et al. (2020)	ResNet-18	52.92%	25.31% ± 0.010	1.6 h

4.12 Additional analysis

In this section, we show more results on different datasets and/or threat models and discuss some implementation details related to the analysis from Sec. 4.5. We also additionally analyze the *smoothness* and *transferability* properties of the models from the Model Zoo.

Progress on adversarial defenses. As done in the main part for the ℓ_∞-robust models on CIFAR-10, we show here the same statistics but for ℓ_2-robust models on CIFAR-10 in Fig. 4.8 and for ℓ_∞-robust models on CIFAR-100 in Fig. 4.9. We observe a few differences compared to the ℓ_∞-robust models on CIFAR-10 reported in Fig. 4.2. First of all, the amount of robustness overestimation is not large and in particular there are no models that have *zero* robust accuracy. Second, we can see that the best ℓ_2-robust models on CIFAR-10 has even higher standard accuracy than a standard model (95.74% vs 94.78%) while having a significantly higher robust accuracy (82.32% vs 0.00%) and leaving a relatively small gap between the standard and robust accuracy. Finally, we note that the progress on the ℓ_∞-threat model on CIFAR-100 is more recent and there are only a few published papers that report adversarial robustness on this dataset.

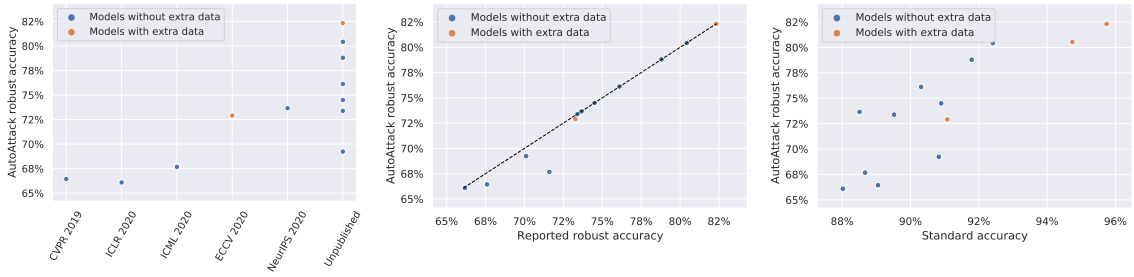


Figure 4.8: Visualization of the robustness and accuracy of 13 CIFAR-10 models from the `RobustBench` ℓ_2-leaderboard. Robustness is evaluated using ℓ_2-perturbations with $\varepsilon_2 = 0.5$.

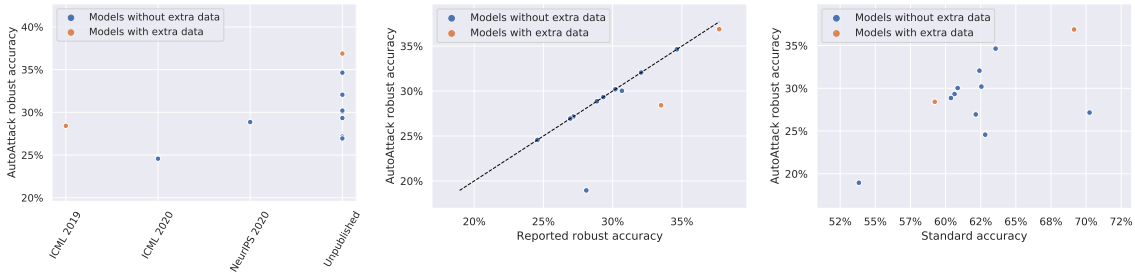


Figure 4.9: Visualization of the robustness and accuracy of 12 CIFAR-100 models from the `RobustBench` ℓ_∞-leaderboard. Robustness is evaluated using ℓ_∞-perturbations with $\varepsilon_\infty = 8/255$.

Robustness across distribution shifts. We measure robust accuracy on various distribution shifts using four dataset, namely CIFAR-10, CINIC-10, CIFAR-10.1, and CIFAR-10-C. In particular, we compute the robust accuracy in the same threat model as for the original CIFAR-10 dataset, and report the results in Fig. 4.10. Interestingly, one can observe that ℓ_p adversarial robustness is maintained under the distribution shifts, and it highly correlates with the robustness on the dataset the models were trained on (i.e. CIFAR-10).

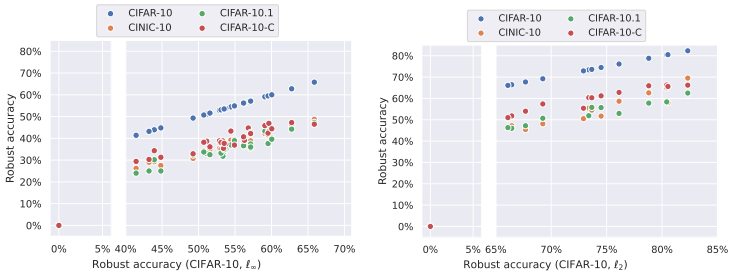


Figure 4.10: Robust accuracy of the robust classifiers, trained against ℓ_∞ and ℓ_2 threat model, respectively, from our Model Zoo on various distribution shifts. The data points with 0% robust accuracy correspond to a standardly trained model.

Calibration. We compute the expected calibration error (ECE) using the code of Guo et al. (2017). We use their default settings to compute the calibration error which includes, in particular, binning of the probability range onto 15 equally-sized bins. However, we use our own implementation of the temperature rescaling algorithm which is close to that

of Augustin et al. (2020). Since optimization of the ECE over the softmax temperature is a simple *one-dimensional* optimization problem, we can solve it efficiently using a grid search. Moreover, the advantage of performing a grid search is that we can optimize directly the metric of interest, i.e. ECE, instead of the cross-entropy loss as in Guo et al. (2017) who had to rely on a differentiable loss since they used LBFGS (Liu and Nocedal, 1989) to optimize the temperature. We perform a grid search over the interval $t \in [0.001, 1.0]$ with a grid step 0.001 and we test both t and $1/t$ temperatures. Moreover, we check that for all models the optimal temperature t is situated not at the boundary of the grid.

We show additional calibration results for ℓ_2-robust models in Fig. 4.11. The overall trend of the ECE is the same as for ℓ_∞-robust models: most of the ℓ_2 models are underconfident (since the optimal temperature is less than one) and lead to worse calibration before and after temperature rescaling. The main difference compared to the ℓ_∞ threat model is that among the ℓ_2 models there are two models that are *better-calibrated*: one before (Engstrom et al. (2019a) with 1.41% ECE vs 3.71% ECE of the standard model) and one after (Gowal et al. (2020) with 1.00% ECE vs 1.11% ECE of the standard model) temperature rescaling. Moreover, we can see that similarly to the ℓ_∞ case, the only overconfident models are either the standard one or models that maximize the margin instead of using norm-bounded perturbations, i.e. Ding et al. (2020) and Rony et al. (2019).

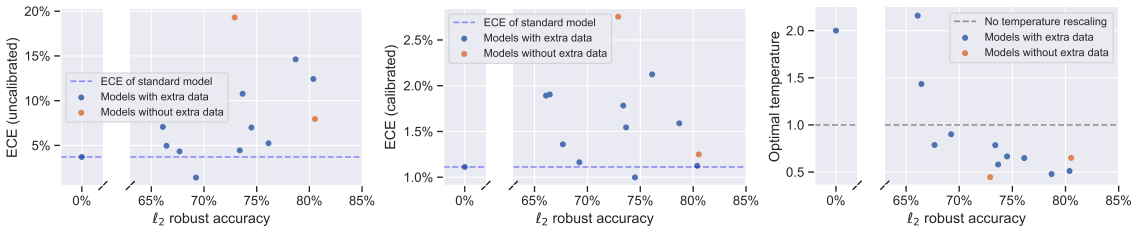


Figure 4.11: Expected calibration error (ECE) before (**left**) and after (**middle**) temperature rescaling, and the optimal rescaling temperature (**right**) for the ℓ_2-robust models.

Out-of-distribution detection. Fig. 4.12 complements Fig. 4.5 and shows the ability of ℓ_2-robust models trained on CIFAR-10 to distinguish inputs from other datasets (CIFAR-100, SVHN, Describable Textures). We find that ℓ_2 robust models have in general comparable OOD detection performance to standardly trained models, while the model by Augustin et al. (2020) achieves even better performance since their approach explicitly optimizes both robust accuracy and worst-case OOD detection performance.

Fairness in robustness. We report the results about fairness for robust models in the ℓ_2-threat model in Fig. 4.13, similarly to what done for ℓ_∞ above. We see that the difference in robustness among classes is similar to what observed for the ℓ_∞ models. Also, the RSD of robustness over classes decreases, which indicates that the disparity among subgroups is reduced, as the average robust accuracy improves. To compute the robustness for the experiments about fairness we used APGD on the targeted DLR loss (Croce and Hein,

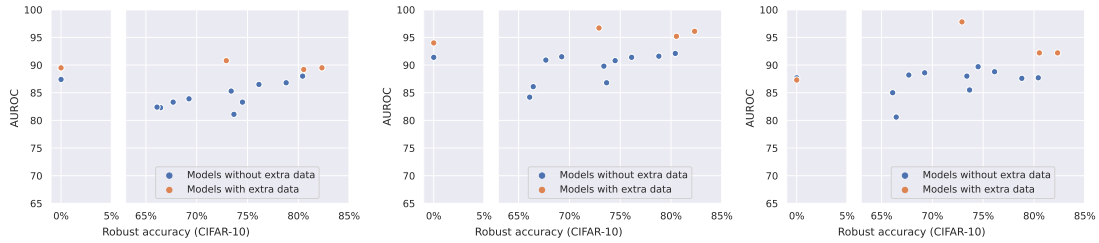


Figure 4.12: Visualization of the quality of OOD detection (higher AUROC is better) for the ℓ_2-robust models on three different OOD datasets: CIFAR-100 (**left**), SVHN (**middle**), Describable Textures (**right**).

2020b) with 3 target classes and 20 iterations each on the whole test set. Note that even with this smaller budget we achieve results very close to that of the full evaluation, with an average difference smaller than 0.5%.

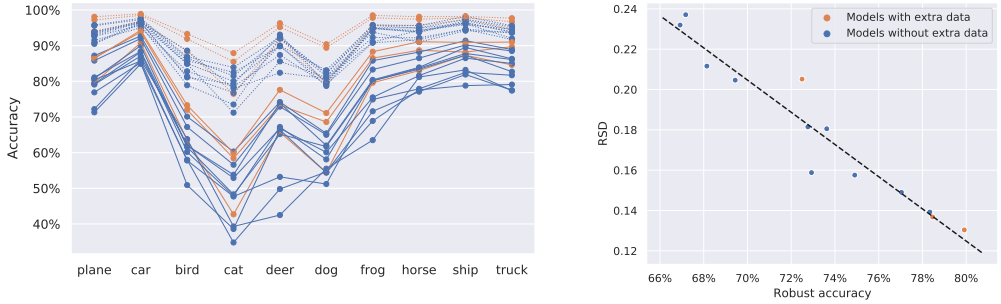


Figure 4.13: Left: classwise standard (dotted lines) and robust (solid) accuracy of ℓ_2-robust models. **Right:** relative standard deviation (RSD) of robust accuracy over classes vs its average.

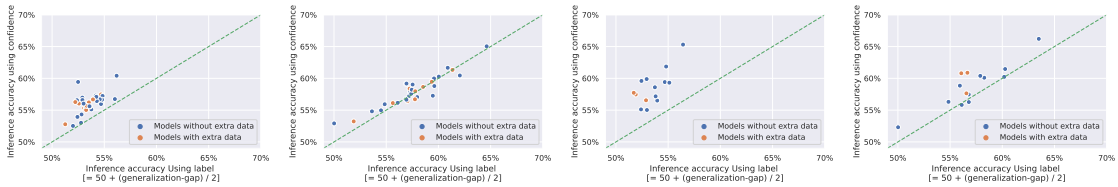
Privacy leakage. We use membership inference accuracy, referred to as inference accuracy, as a measure of the leakage of training data details from pre-trained neural networks. It measures how successfully we can identify whether a particular sample was present in the training set. We closely follow the methodology described in Song and Mittal (2021) to calculate inference accuracy. In particular, we measure the confidence in the correct class for each input image with a pre-trained classifier. We measure the confidence for both training and test set images and calculate the maximum classification accuracy between train and test images based on the confidence values. We refer to this accuracy as *inference accuracy using confidence.* We also follow the recommendation from Song et al. (2019) where they show that adversarial examples are more successful in estimating inference accuracy on robust networks. In our experiments, we also find that using adversarial examples leads to higher inference accuracy than benign images (Figure 4.14). We also find that robust networks in the ℓ_2 threat model have relatively higher inference accuracy than robust networks in the ℓ_∞ threat model.

A key reason behind privacy leakage through membership inference is that deep neural networks often end up overfitting on the training data. One standard metric to measure overfitting is the generalization gap between train and test set. Naturally, this difference in the accuracy on the train and test set is the baseline of inference accuracy. We refer to

it as *inference accuracy using label* and report it in Figure 4.15. We consider both benign and adversarial images. When using benign images, we find confidence information does lead to higher inference accuracy than using only labels. However, with adversarial examples, which achieve higher inference accuracy than benign images, we find that inference accuracy based on confidence information closely follows the inference accuracy calculate from labels.

(a) Benign (ℓ_∞) **(b)** Adversarial (ℓ_∞) **(c)** Benign (ℓ_2) **(d)** Adversarial (ℓ_2)

Figure 4.14: Comparing privacy leakage of different networks. We compare membership inference accuracy from benign and adversarial images across both ℓ_∞ and ℓ_2 threat model.



(a) Benign (ℓ_∞) **(b)** Adversarial (ℓ_∞) **(c)** Benign (ℓ_2) **(d)** Adversarial (ℓ_2)

Figure 4.15: Comparing privacy leakage with different output statistics. We measure privacy leakage using membership inference accuracy, i.e., classification success between train and test set. We measure it using two baselines 1) based on correct prediction i.e., using predicted class label and 2) based on classification confidence in correct class. We also measure it using both benign and adversarial images.

Smoothness. Previous work Yang et al. (2020) has shown that smoothness of a model, together with enough separation between the classes of the dataset for which it is trained, is necessary to achieve both natural and robust accuracy. They use local Lipschitzness as a measure for model smoothness, and observe empirically that robust models are more smoother than models trained in a standard way. Our Model Zoo enables us to check this fact empirically on a wider range of robust models, trained with a more diverse set of techniques, in particular with and without extra training data. Moreover, as we have access to the model internals, we can also compute local Lipschitzness of the model up to arbitrary layers, to see how smoothness changes between layers.

We compute local Lipschitzness using projected gradient descent (PGD) on the following optimization problem:

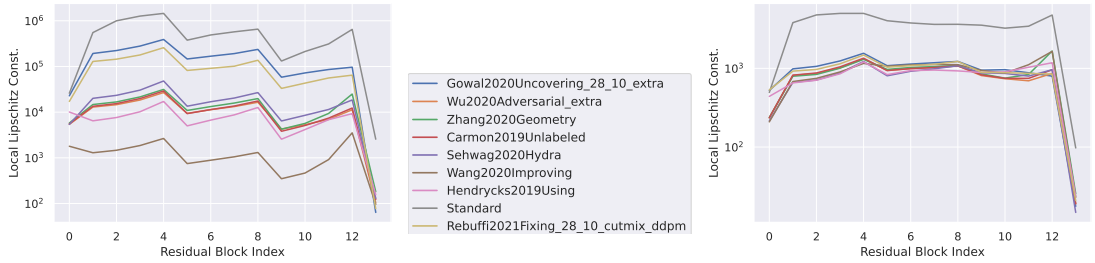
$$L = \frac{1}{N}\sum_{i=1}^{N} \max_{\substack{x_1:\|x_1-x_i\|_\infty\leq\varepsilon,\\ x_2:\|x_2-x_i\|_\infty\leq\varepsilon}} \frac{\|f(x_1)-f(x_2)\|_1}{\|x_1-x_2\|_\infty}, \tag{4.2}$$

where x_i represents each sample around which we compute local Lipschitzness, N is the number of samples across which we average ($N = 256$ in all our experiments), and f represents the function whose Lipschitz constant we compute. As mentioned above, this function can be either the full model, or the model up to an arbitrary intermediate layer.

Since the models can have similar smoothness behavior, but at a different scale, we also consider normalizing the models outputs. One such normalization we use is given by the projection of the model outputs on the unit ℓ_2 ball. This normalization aims at capturing the angular change of the output, instead of taking in consideration also its magnitude. We compute the "angular" version of the Lipschitz constant as

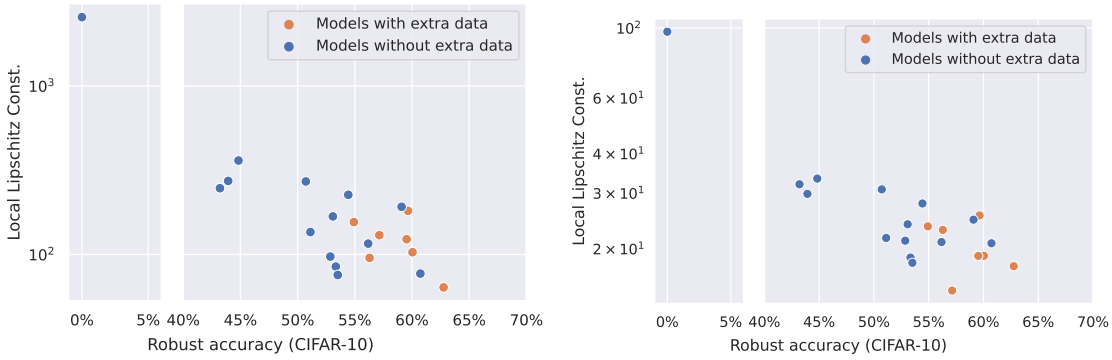
$$L = \frac{1}{N}\sum_{i=1}^{N} \max_{\substack{x_1:\|x_1-x_i\|_\infty\leq\varepsilon,\\ x_2:\|x_2-x_i\|_\infty\leq\varepsilon}} \frac{\left\| \frac{f(x_1)}{\|f(x_1)\|_2} - \frac{f(x_2)}{\|f(x_2)\|_2} \right\|_1}{\|x_1 - x_2\|_\infty}. \tag{4.3}$$

For both variations of Lipschitzness, we compute it with $\varepsilon = 8/255$, running the PDG-like procedure for 50 steps, with a step size of $\varepsilon/5$.



(a) No normalization **(b)** ℓ_2 unit ball normalization

Figure 4.16: Lipschitzness. Computation of the local Lipschitz constant of the WRN-28-10 ℓ_∞-robust models in our Model Zoo with $\varepsilon = 8/255$. The color coding of the models is the same across both figures. For the correspondence between model IDs (shown in the legend) and papers that introduced them, see Appendix 4.13.



(a) $\varepsilon = 8/255$, no normalization **(b)** $\varepsilon = 8/255$, ℓ_2 unit ball normalization

Figure 4.17: Lipschitzness vs Robustness. Local Lipschitz constant of the output layer vs. robust accuracy of a subset of the ℓ_∞-robust models in our Model Zoo.

In Fig. 4.16 we compute the layerwise Lipschitzness for all ℓ_∞ models trained on CIFAR-10 from the Model Zoo that have the WRN-28-10 architecture. We observe that the standard model is the least smooth at all the layers, and that all the robustly trained models are smoother. Moreover, we can notice that in Fig. 4.16a there are two models in the middle ground: these are the models by Gowal et al. (2020) and Rebuffi et al. (2021), which are the most robust ones, up to the last layer, the smoothest. Nonetheless, in the middle layers, they are the second and third least smooth, according to the unnormalized local Lipschitzness. This can be due to the different activation function used in these models (Swish vs ReLU). For this reason, we also compute "angular" Lipschitzness according to Eq. 4.3. Indeed, in Fig. 4.16b, all the robust models are in the same order of magnitude at all layers.

Finally, we also show the Lipschitz constants of the output layer for a larger set of ℓ_∞ models from the Model Zoo that are not restricted to the same architecture. We plot the Lipschitz constant vs. the robust accuracy for these models in Fig 4.17. We see that there is a clear relationship between robust accuracy and Lipschitzness, hence confirming the findings of Yang et al. (2020).

Transferability. We generate adversarial examples for a network, referred to as source network, and measure robust accuracy of every other network, referred to as target network, from the model zoo on them. We also include additional non-robust models[10], to name a few, VGG19, ResNet18, and DenseNet121, in our analysis. We consider both ten step PGD attack and FGSM attack to generate adversarial examples as two transferability baselines commonly used in the literature. For both attacks, we use the cross-entropy loss, and for the PGD attack we use ten iterations and step size $\varepsilon/4$.

We present our results in Figure 4.18, 4.19 where the correspondence between model IDs and papers that introduced them can be found in Appendix 4.13. We find that transferability to each robust target network follows a similar trend where adversarial examples transfer equally well from another robust networks. Though slight worse than robust network, adversarial example from non-robust network also transfer equally well to robust networks. We observe a strong transferability among non-robust networks with adversarial examples generated from PGD attacks. Adversarial examples generated using the FGSM attack also transfer successfully. However, they achieve lower robust accuracy on the target network. Intriguingly, we observe the weakest transferability from a robust to a non-robust network. This observation holds for all robust source networks across both FGSM and PGD-attack in both ℓ_∞ and ℓ_2 threat model.

4.13 Leaderboards

We here report the details of all the models included in the various leaderboards, for the ℓ_∞-, ℓ_2-threat models and common corruptions. In particular, we show for each model

[10]We train then for 200 epochs and achieve 93-95% clean accuracy for all networks on the CIFAR-10 dataset.

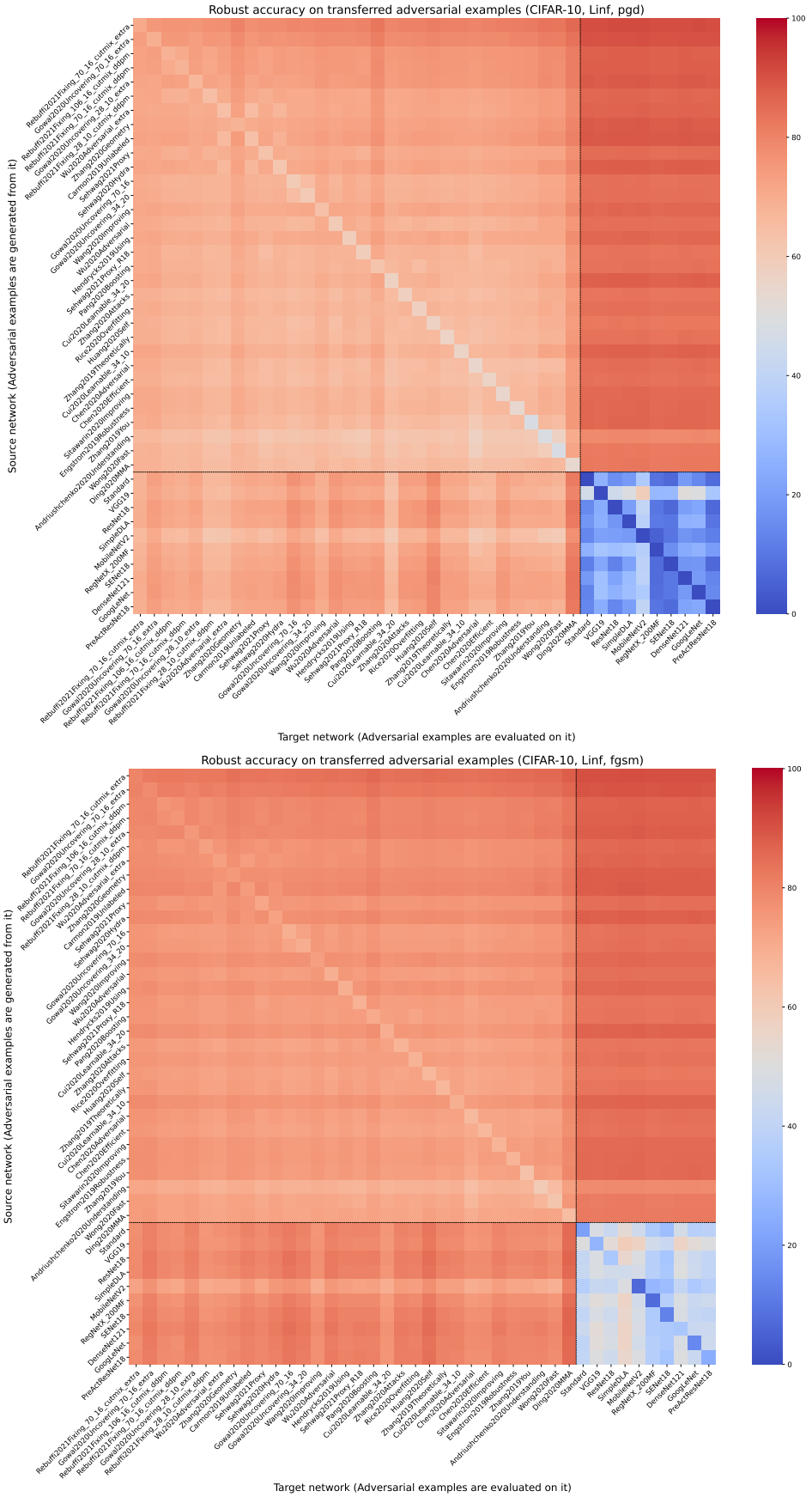


Figure 4.18: Measuring transferability of adversarial examples (ℓ_∞, $\epsilon = 8/255$). We use a ten step PGD attack in top figure and FGSM attack in bottom figure. Lower robust accuracy implies better transferability.

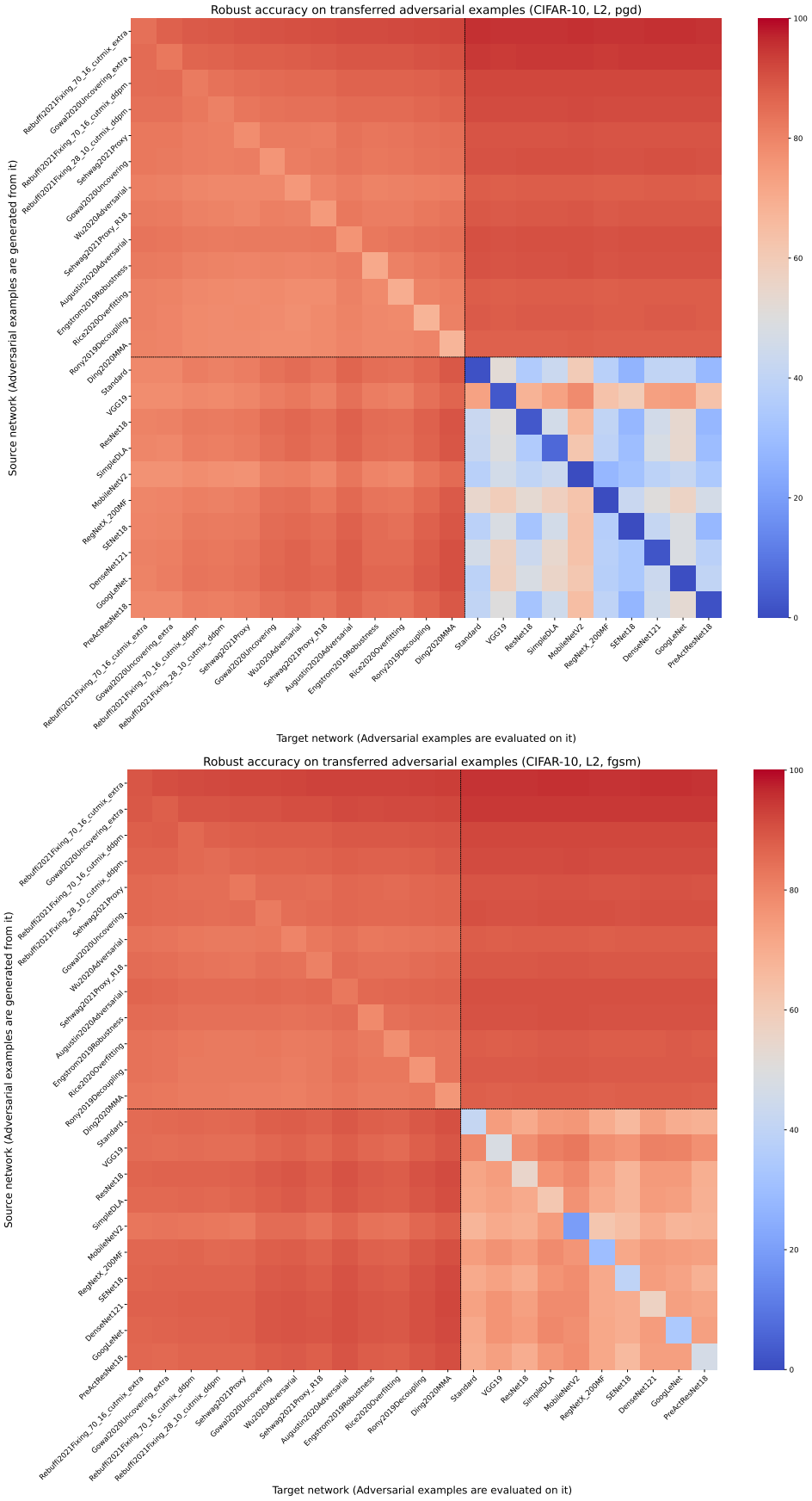


Figure 4.19: Measuring transferability of adversarial examples (ℓ_2, $\epsilon = 0.5$). We use a ten step PGD attack in top figure and FGSM attack in bottom figure. Lower robust accuracy implies better transferability.

the clean accuracy, robust accuracy (either on adversarial attacks or corrupted images), whether additional data is used for training, the architecture used, the venue at which it appeared and, if available, the identifier in the Model Zoo (which is also used in some of the experiments in Sec. 4.12).

Table 4.3: Leaderboard for the ℓ_∞-threat model, CIFAR-10.

	Model	Clean	Robust	Extra data	Architecture	Venue	Model Zoo ID
1	Rebuffi et al. (2021)	92.23	66.56	Y	WRN-70-16	arXiv, Mar 2021	rebuffi2021fixing_70_16_cutmix_extra
2	Gowal et al. (2020)	91.10	65.87	Y	WRN-70-16	arXiv, Oct 2020	Gowal2020Uncovering_70_16_extra
3	Rebuffi et al. (2021)	88.50	64.58	N	WRN-106-16	arXiv, Mar 2021	rebuffi2021fixing_106_16_cutmix_ddpm
4	Rebuffi et al. (2021)	88.54	64.20	N	WRN-70-16	arXiv, Mar 2021	rebuffi2021fixing_70_16_cutmix_ddpm
5	Rade and Moosavi-Dezfooli (2021)	91.47	62.83	Y	WRN-34-10	OpenReview, Jun 2021	Rade2021Helper_extra
6	Gowal et al. (2020)	89.48	62.76	Y	WRN-28-10	arXiv, Oct 2020	Gowal2020Uncovering_28_10_extra
7	Rade and Moosavi-Dezfooli (2021)	88.16	60.97	N	WRN-28-10	OpenReview, Jun 2021	Rade2021Helper_ddpm
8	Rebuffi et al. (2021)	87.33	60.73	N	WRN-28-10	arXiv, Mar 2021	rebuffi2021fixing_28_10_cutmix_ddpm
9	Wu et al. (2020a)	87.67	60.65	Y	WRN-34-15	arXiv, Oct 2020	N/A
10	Sridhar et al. (2021)	86.53	60.41	Y	WRN-34-15	arXiv, Jun 2021	Sridhar2021Robust_34_15
11	Wu et al. (2020b)	88.25	60.04	Y	WRN-28-10	NeurIPS 2020	Wu2020Adversarial_extra
12	Sridhar et al. (2021)	89.46	59.66	Y	WRN-28-10	arXiv, Jun 2021	Sridhar2021Robust
13	Zhang et al. (2021a)	89.36	59.64	Y	WRN-28-10	ICLR 2021	Zhang2020Geometry
14	Carmon et al. (2019)	89.69	59.53	Y	WRN-28-10	NeurIPS 2019	carmon2019unlabeled
15	Sehwag et al. (2021)	85.85	59.09	N	WRN-34-10	arXiv, Apr 2021	Sehwag2021Proxy
16	Rade and Moosavi-Dezfooli (2021)	89.02	57.67	Y	PreActRN-18	OpenReview, Jun 2021	Rade2021Helper_R18_extra
17	Gowal et al. (2020)	85.29	57.14	N	WRN-70-16	arXiv, Oct 2020	Gowal2020Uncovering_70_16
18	Sehwag et al. (2020a)	88.98	57.14	Y	WRN-28-10	NeurIPS 2020	Sehwag2020Hydra
19	Rade and Moosavi-Dezfooli (2021)	86.86	57.09	N	PreActRN-18	OpenReview, Jun 2021	Rade2021Helper_R18_ddpm
20	Gowal et al. (2020)	85.64	56.82	N	WRN-34-20	arXiv, Oct 2020	Gowal2020Uncovering_34_20
21	Rebuffi et al. (2021)	83.53	56.66	N	PreActRN-18	arXiv, Mar 2021	rebuffi2021fixing_R18_ddpm
22	Wang et al. (2020)	87.50	56.29	Y	WRN-28-10	ICLR 2020	Wang2020Improving
23	Wu et al. (2020b)	85.36	56.17	N	WRN-34-10	NeurIPS 2020	Wu2020Adversarial
24	Uesato et al. (2019)	86.46	56.03	Y	WRN-28-10	NeurIPS 2019	N/A
25	Hendrycks et al. (2019)	87.11	54.92	Y	WRN-28-10	ICML 2019	hendrycks2019using
26	Sehwag et al. (2021)	84.38	54.43	N	RN-18	arXiv, Apr 2021	Sehwag2021Proxy_R18
27	Pang et al. (2021a)	86.43	54.39	N	WRN-34-20	ICLR 2021	N/A
28	Pang et al. (2020c)	85.14	53.74	N	WRN-34-20	NeurIPS 2020	Pang2020Boosting
29	Cui et al. (2021)	88.70	53.57	N	WRN-34-20	ICCV 2021	Cui2020Learnable_34_20
30	Zhang et al. (2020b)	84.52	53.51	N	WRN-34-10	ICML 2020	Zhang2020Attacks
31	Rice et al. (2020)	85.34	53.42	N	WRN-34-20	ICML 2020	rice2020overfitting
32	Huang et al. (2020)	83.48	53.34	N	WRN-34-10	NeurIPS 2020	Huang2020Self
33	Zhang et al. (2019c)	84.92	53.08	N	WRN-34-10	ICML 2019	zhang2019theoretically
34	Cui et al. (2021)	88.22	52.86	N	WRN-34-10	ICCV 2021	Cui2020Learnable_34_10
35	Qin et al. (2019)	86.28	52.84	N	WRN-40-8	NeurIPS 2019	N/A
36	Chen et al. (2020b)	86.04	51.56	N	RN-50	CVPR 2020	Chen2020Adversarial
37	Chen et al. (2020a)	85.32	51.12	N	WRN-34-10	arXiv, Oct 2020	Chen2020Efficient
38	Sitawarin et al. (2020)	86.84	50.72	N	WRN-34-10	arXiv, Mar 2020	Sitawarin2020Improving
39	Engstrom et al. (2019a)	87.03	49.25	N	RN-50	GitHub, Oct 2019	Engstrom2019Robustness
40	Singh et al. (2019)	87.80	49.12	N	WRN-34-10	IJCAI 2019	N/A
41	Mao et al. (2019)	86.21	47.41	N	WRN-34-10	NeurIPS 2019	N/A
42	Zhang et al. (2019a)	87.20	44.83	N	WRN-34-10	NeurIPS 2019	zhang2019propagate
43	Madry et al. (2018)	87.14	44.04	N	WRN-34-10	ICLR 2018	N/A
44	Andriushchenko and Flammarion (2020)	79.84	43.93	N	PreActRN-18	NeurIPS 2020	Andriushchenko2020Understanding
45	Pang et al. (2020a)	80.89	43.48	N	RN-32	ICLR 2020	N/A
46	Wong et al. (2020)	83.34	43.21	N	PreActRN-18	ICLR 2020	wong2020fast
47	Shafahi et al. (2019)	86.11	41.47	N	WRN-34-10	NeurIPS 2019	N/A
48	Ding et al. (2020)	84.36	41.44	N	WRN-28-4	ICLR 2020	Ding2020MMA
49	Kundu et al. (2021)	87.32	40.41	N	RN-18	ASP-DAC 2021	N/A
50	Atzmon et al. (2019)	81.30	40.22	N	RN-18	NeurIPS 2019	N/A
51	Moosavi-Dezfooli et al. (2019a)	83.11	38.50	N	RN-18	CVPR 2019	N/A
52	Zhang and Wang (2019)	89.98	36.64	N	WRN-28-10	NeurIPS 2019	N/A
53	Zhang and Xu (2019)	90.25	36.45	N	WRN-28-10	OpenReview, Sep 2019	N/A
54	Jang et al. (2019)	78.91	34.95	N	RN-20	ICCV 2019	N/A
55	Kim and Wang (2019)	91.51	34.22	N	WRN-34-10	OpenReview, Sep 2019	N/A
56	Zhang et al. (2020a)	44.73	32.64	N	5-layer-CNN	ICLR 2020	N/A
57	Wang and Zhang (2019)	92.80	29.35	N	WRN-28-10	ICCV 2019	N/A
58	Xiao et al. (2020)	79.28	7.15	N	DenseNet-121	ICLR 2020	N/A
59	Jin and Rinard (2020)	90.84	1.35	N	RN-18	arXiv, Mar 2020	N/A
60	Mustafa et al. (2019)	89.16	0.28	N	RN-110	ICCV 2019	N/A
61	Chan et al. (2020)	93.79	0.26	N	WRN-34-10	ICLR 2020	N/A
62	Standard	94.78	0.0	N	WRN-28-10	N/A	N/A
63	Alfarra et al. (2020)	91.03	0.00	N	WRN-28-10	arXiv, Jun 2020	N/A

Table 4.4: Leaderboard for the ℓ_2-threat model, CIFAR-10.

	Model	Clean	Robust	Extra data	Architecture	Venue	Model Zoo ID
1	Rebuffi et al. (2021)	95.74	82.32	Y	WRN-70-16	arXiv, Mar 2021	rebuffi2021fixing_70_16_cutmix_extra
2	Gowal et al. (2020)	94.74	80.53	Y	WRN-70-16	arXiv, Oct 2020	Gowal2020Uncovering_extra
3	Rebuffi et al. (2021)	92.41	80.42	N	WRN-70-16	arXiv, Mar 2021	rebuffi2021fixing_70_16_cutmix_ddpm
4	Rebuffi et al. (2021)	91.79	78.80	N	WRN-28-10	arXiv, Mar 2021	rebuffi2021fixing_28_10_cutmix_ddpm
5	Augustin et al. (2020)	93.96	78.79	Y	WRN-34-10	ECCV 2020	Augustin2020Adversarial_34_10_extra
6	Augustin et al. (2020)	92.23	76.25	Y	WRN-34-10	ECCV 2020	Augustin2020Adversarial_34_10
7	Rade and Moosavi-Dezfooli (2021)	90.57	76.15	N	PreActRN-18	OpenReview, Jun 2021	Rade2021Helper_R18_ddpm
8	Sehwag et al. (2021)	90.31	76.12	N	WRN-34-10	arXiv, Apr 2021	Sehwag2021Proxy
9	Rebuffi et al. (2021)	90.33	75.86	N	PreActRN-18	arXiv, Mar 2021	rebuffi2021fixing_R18_cutmix_ddpm
10	Gowal et al. (2020)	90.90	74.50	N	WRN-70-16	arXiv, Oct 2020	Gowal2020Uncovering
11	Wu et al. (2020b)	88.51	73.66	N	WRN-34-10	NeurIPS 2020	Wu2020Adversarial
12	Sehwag et al. (2021)	89.52	73.39	N	RN-18	arXiv, Apr 2021	Sehwag2021Proxy_R18
13	Augustin et al. (2020)	91.08	72.91	Y	RN-50	ECCV 2020	Augustin2020Adversarial
14	Engstrom et al. (2019a)	90.83	69.24	N	RN-50	GitHub, Sep 2019	Engstrom2019Robustness
15	Rice et al. (2020)	88.67	67.68	N	PreActRN-18	ICML 2020	rice2020overfitting
16	Rony et al. (2019)	89.05	66.44	N	WRN-28-10	CVPR 2019	Rony2019Decoupling
17	Ding et al. (2020)	88.02	66.09	N	WRN-28-4	ICLR 2020	Ding2020MMA
18	Standard	94.78	0.0	N	WRN-28-10	N/A	Standard

Table 4.5: Leaderboard for common corruptions, CIFAR-10.

	Model	Clean	Corr.	Extra data	Architecture	Venue	Model Zoo ID
1	Calian et al. (2021)	94.93	92.17	Y	RN-50	arXiv, Apr 2021	N/A
2	Kireev et al. (2021)	94.75	89.60	N	RN-18	arXiv, Mar 2021	Kireev2021Effectiveness_RLATAugMix
3	Hendrycks et al. (2020)	95.83	89.09	N	ResNeXt29_32x4d	ICLR 2020	Hendrycks2020AugMix_ResNeXt
4	Hendrycks et al. (2020)	95.08	88.82	N	WRN-40-2	ICLR 2020	Hendrycks2020AugMix_WRN
5	Kireev et al. (2021)	94.77	88.53	N	PreActRN-18	arXiv, Mar 2021	Kireev2021Effectiveness_RLATAugMixNoJSD
6	Rebuffi et al. (2021)	92.23	88.23	Y	WRN-70-16	arXiv, Mar 2021	rebuffi2021fixing_70_16_cutmix_extra_L2
7	Gowal et al. (2020)	94.74	87.68	Y	WRN-70-16	arXiv, Oct 2020	N/A
8	Kireev et al. (2021)	94.97	86.60	N	PreActRN-18	arXiv, Mar 2021	Kireev2021Effectiveness_AugMixNoJSD
9	Kireev et al. (2021)	93.24	85.04	N	PreActRN-18	arXiv, Mar 2021	Kireev2021Effectiveness_Gauss50percent
10	Gowal et al. (2020)	90.90	84.90	N	WRN-70-16	arXiv, Oct 2020	N/A
11	Kireev et al. (2021)	93.10	84.10	N	PreActRN-18	arXiv, Mar 2021	Kireev2021Effectiveness_RLAT
12	Rebuffi et al. (2021)	92.23	82.82	Y	WRN-70-16	arXiv, Mar 2021	rebuffi2021fixing_70_16_cutmix_extra_Linf
13	Gowal et al. (2020)	91.10	81.84	Y	WRN-70-16	arXiv, Oct 2020	N/A
14	Gowal et al. (2020)	85.29	76.37	N	WRN-70-16	arXiv, Oct 2020	N/A
15	Standard	94.78	73.46	N	WRN-28-10	N/A	Standard

Table 4.6: Leaderboard for the ℓ_∞-threat model, CIFAR-100.

	Model	Clean	Robust	Extra data	Architecture	Venue	Model Zoo ID
1	Gowal et al. (2020)	69.15	36.88	Y	WRN-70-16	arXiv, Oct 2020	Gowal2020Uncovering_extra
2	Rebuffi et al. (2021)	63.56	34.64	N	WRN-70-16	arXiv, Mar 2021	rebuffi2021fixing_70_16_cutmix_ddpm
3	Rebuffi et al. (2021)	62.41	32.06	N	WRN-28-10	arXiv, Mar 2021	rebuffi2021fixing_28_10_cutmix_ddpm
4	Cui et al. (2021)	62.55	30.20	N	WRN-34-20	ICCV 2021	Cui2020Learnable_34_20_LBGAT6
5	Gowal et al. (2020)	60.86	30.03	N	WRN-70-16	arXiv, Oct 2020	Gowal2020Uncovering
6	Cui et al. (2021)	60.64	29.33	N	WRN-34-10	ICCV 2021	Cui2020Learnable_34_10_LBGAT6
7	Rade and Moosavi-Dezfooli (2021)	61.50	28.88	N	PreActRN-18	OpenReview, Jun 2021	Rade2021Helper_R18_ddpm
8	Wu et al. (2020b)	60.38	28.86	N	WRN-34-10	NeurIPS 2020	Wu2020Adversarial
9	Rebuffi et al. (2021)	56.87	28.50	N	PreActRN-18	arXiv, Mar 2021	rebuffi2021fixing_R18_ddpm
10	Hendrycks et al. (2019)	59.23	28.42	Y	WRN-28-10	ICML 2019	hendrycks2019using
11	Cui et al. (2021)	70.25	27.16	N	WRN-34-10	ICCV 2021	Cui2020Learnable_34_10_LBGAT0
12	Chen et al. (2020a)	62.15	26.94	N	WRN-34-10	arXiv, Oct 2020	Chen2020Efficient
13	Sitawarin et al. (2020)	62.82	24.57	N	WRN-34-10	ICML 2020	Sitawarin2020Improving
14	Rice et al. (2020)	53.83	18.95	N	PreActRN-18	ICML 2020	rice2020overfitting

Table 4.7: Leaderboard for common corruptions, CIFAR-100.

	Model	Clean	Corr.	Extra data	Architecture	Venue	Model Zoo ID
1	Hendrycks et al. (2020)	78.90	65.14	N	ResNeXt29_32x4d	ICLR 2020	Hendrycks2020AugMix_ResNeXt
2	Hendrycks et al. (2020)	76.28	64.11	N	WRN-40-2	ICLR 2020	Hendrycks2020AugMix_WRN
3	Gowal et al. (2020)	69.15	56.00	Y	WRN-70-16	arXiv, Oct 2020	Gowal2020Uncovering_extra_Linf
4	Gowal et al. (2020)	60.86	49.46	N	WRN-70-16	arXiv, Oct 2020	Gowal2020Uncovering_Linf

Table 4.8: Leaderboard for the ℓ_∞-threat model, ImageNet.

	Model	Clean	Robust	Extra data	Architecture	Venue	Model Zoo ID
1	Salman et al. (2020)	68.46	38.14	N	WRN-50-2	NeurIPS 2020	Salman2020Do_50_2
2	Salman et al. (2020)	64.02	34.96	N	RN-50	NeurIPS 2020	Salman2020Do_R50
3	Engstrom et al. (2019a)	62.56	29.22	N	RN-50	GitHub, Oct 2019	Engstrom2019Robustness
4	Wong et al. (2020)	55.62	26.24	N	RN-18	ICLR 2020	wong2020fast
5	Salman et al. (2020)	52.92	25.32	N	RN-50	NeurIPS 2020	Salman2020Do_R18
6	Standard_R50	76.52	0.0	N	RN-50	N/A	Standard_R50

Table 4.9: Leaderboard for common corruptions, ImageNet.

	Model	Clean	Corr.	Extra data	Architecture	Venue	Model Zoo ID
1	Hendrycks et al. (2021a)	76.88	51.61	N	RN-50	ICCV 2021	Hendrycks2020Many
2	Hendrycks et al. (2020)	76.98	46.91	N	RN-50	ICLR 2020	Hendrycks2020AugMix
3	Geirhos et al. (2019)	74.88	44.48	N	RN-50	ICLR 2019	Geirhos2018_SIN_IN
4	Geirhos et al. (2019)	77.44	40.77	N	RN-50	ICLR 2019	Geirhos2018_SIN_IN_IN
5	Standard_R50	76.52	38.12	N	RN-50	N/A	Standard_R50
6	Geirhos et al. (2019)	60.24	37.95	N	RN-50	ICLR 2019	Geirhos2018_SIN
7	Salman et al. (2020)	68.46	34.60	N	WRN-50-2	NeurIPS 2020	Salman2020Do_50_2_Linf

Part II
Generalization in Modern Deep Learning

5 Towards Understanding Sharpness-Aware Minimization

5.1 Preface

In this chapter, based on Andriushchenko and Flammarion (2022), we focus on the recent training method called Sharpness-Aware Minimization (SAM) which has been shown to significantly improve generalization in various settings. We argue that the existing justifications for the success of SAM which are based on a PAC-Bayes generalization bound and the idea of convergence to flat minima are incomplete. Via a set of experiments and theoretical results, we provide a new perspective on the reasons behind better generalization of SAM.

Summary Sharpness-Aware Minimization (SAM) is a recent training method that relies on worst-case weight perturbations which significantly improves generalization in various settings. We argue that the existing justifications for the success of SAM which are based on a PAC-Bayes generalization bound and the idea of convergence to flat minima are incomplete. Moreover, there are no explanations for the success of using m-sharpness in SAM which has been shown as essential for generalization. To better understand this aspect of SAM, we theoretically analyze its implicit bias for diagonal linear networks. We prove that SAM always chooses a solution that enjoys better generalization properties than standard gradient descent for a certain class of problems, and this effect is amplified by using m-sharpness. We further study the properties of the implicit bias on non-linear networks empirically, where we show that fine-tuning a standard model with SAM can lead to significant generalization improvements. Finally, we provide convergence results of SAM for non-convex objectives when used with stochastic gradients. We illustrate these results empirically for deep networks and discuss their relation to the generalization behavior of SAM. The code of our experiments is available at https://github.com/tml-epfl/understanding-sam.

Co-authors Nicolas Flammarion.

Contributions Maksym Andriushchenko made key contributions to all aspects of the project.

5.2 Introduction

Understanding generalization of overparametrized deep neural networks is a central topic of machine learning. Training objective has many global optima where the training data are perfectly fitted (Zhang et al., 2017a), but different global optima lead to dramatically different generalization performance (Liu et al., 2019). However, it has been observed that stochastic gradient descent (SGD) tends to converge to well-generalizing solutions, even *without* any explicit regularization methods (Zhang et al., 2017a). This suggests that the leading role is played by the *implicit* bias of the optimization algorithms used (Neyshabur et al., 2015): when the training objective is minimized using a particular algorithm and initialization method, it converges to a specific solution with favorable generalization properties. However, even though SGD has a very beneficial implicit bias, significant overfitting can still occur, particularly in the presence of label noise (Nakkiran et al., 2020) and adversarial perturbations (Rice et al., 2020).

Recently it has been observed that the *sharpness* of the training loss, i.e., how quickly it changes in some neighborhood around the parameters of the model, correlates well with the generalization error (Keskar et al., 2016; Jiang et al., 2019), and generalization bounds related to the sharpness have been derived (Dziugaite and Roy, 2018). The idea of minimizing the sharpness to improve generalization has motivated recent works of Foret et al. (2021), Zheng et al. (2021), and Wu et al. (2020b) which propose to use worst-case perturbations of the weights on every iteration of training in order to improve generalization. We refer to this method as *Sharpness-Aware Minimization* (SAM) and focus mainly on the version proposed in Foret et al. (2021) that performs only one step of gradient ascent to approximately solve the weight perturbation problem before updating the weights.

Despite the fact that SAM significantly improves generalization in various settings, the existing justifications based on the generalization bounds provided by Foret et al. (2021) and Wu et al. (2020b) do not seem conclusive. The main reason is that their generalization bounds do not distinguish the robustness to *worst-case* weight perturbation from *average-case* robustness to Gaussian noise. However the latter does not sufficiently improve generalization as both Foret et al. (2021) and Wu et al. (2020b) report. Furthermore, their analysis does not distinguish whether the worst-case weight perturbation is computed based on some or on all training examples. As we will discuss, this feature has a crucial impact on generalization.

In our paper, we aim to further investigate the reasons for SAM's success and make the following contributions:

- We discuss why the current understanding of the success of SAM which is based on a PAC-Bayesian generalization bound and on convergence to a flatter minimum is incomplete.
- We test hypotheses regarding why maximization in SAM taken over *fewer* training points can lead to better generalization and conclude that the benefit is likely to

come from the better objective.

- We study the implicit bias of this objective theoretically for diagonal linear networks. For non-linear networks, we study the implicit bias empirically and relate it to the theoretical model.

- We prove convergence of SAM for non-convex objectives in the stochastic setting. We check convergence empirically for deep networks and relate it to the generalization behavior of SAM.

5.3 Background on SAM

Related work. Here we discuss relevant works on robustness in the *weight space* and its relation to generalization. Works on weight-space robustness of neural networks date back at least to the 1990s (Murray and Edwards, 1993; Hochreiter and Schmidhuber, 1995). Random perturbations of the weights are used extensively in deep learning (Jim et al., 1996; Graves et al., 2013), and most prominently in approaches such as dropout (Srivastava et al., 2014). Many practitioners have observed that using SGD with larger batches for training leads to worse generalization (LeCun et al., 2012), and Keskar et al. (2016) have shown that this degradation of performance is *correlated* with the sharpness of the found parameters. This observation has motivated many further works which focus on closing the generalization gap between small-batch and large-batch SGD (Wen et al., 2018; Haruki et al., 2019; Lin et al., 2020). More recently, Jiang et al. (2019) have shown a strong correlation between the sharpness and the generalization error on a large set of models under a variety of different settings hyperparameters, beyond the batch size. This has motivated the idea of minimizing the sharpness during training to improve standard generalization, leading to Sharpness-Aware Minimization (SAM) (Foret et al., 2021). SAM modifies SGD such that on every iteration of training, the gradient is taken not at the current iterate but rather at a worst-case point in its vicinity. Zheng et al. (2021) concurrently propose a similar weight perturbation method which also successfully improves standard generalization on multiple deep learning benchmarks. Wu et al. (2020b) have also proposed an almost identical algorithm with the same motivation, but with the focus on improving robust generalization of adversarial training. On the theoretical side, Mulayoff and Michaeli (2020) study the sharpness properties of minima of deep linear network, and Neu (2021); Wang and Mao (2022) study generalization bounds based on average-case sharpness and quantities related to the optimization trajectory of SGD.

Sharpness. Let $\mathcal{S}_{train} = \{x_i, y_i\}_{i=1}^n$ be the training data and $\ell_i(\boldsymbol{w})$ be the loss of a classifier parametrized by weights $w \in \mathbb{R}^{|w|}$ and evaluated at point (x_i, y_i). Then the *sharpness* on a set of points $\mathcal{S} \subseteq \mathcal{S}_{train}$ is defined as:

$$s(\boldsymbol{w}, \mathcal{S}) \triangleq \max_{\|\boldsymbol{\delta}\|_2 \leq \rho} \frac{1}{|\mathcal{S}|} \sum_{i:(x_i,y_i)\in\mathcal{S}} \ell_i(\boldsymbol{w}+\boldsymbol{\delta}) - \ell_i(\boldsymbol{w}). \tag{5.1}$$

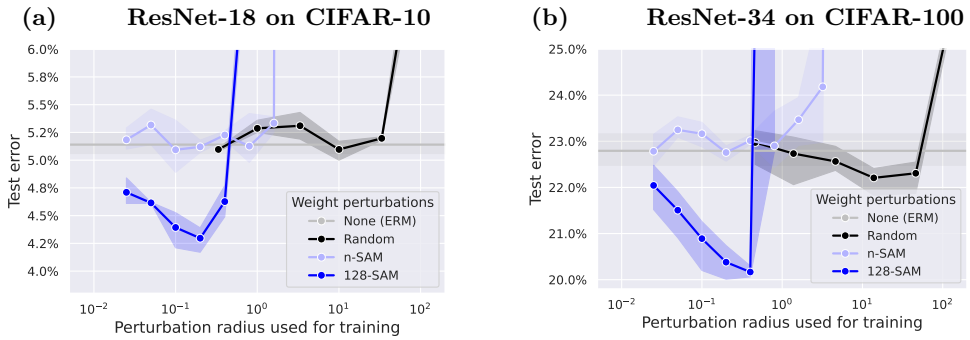


Figure 5.1: Comparison of different weight perturbation methods: no perturbations (ERM), random perturbations prior to taking the gradient on each iteration, n-SAM, and 128-SAM (see Sec. 5.3 for the notation). All models are trained with standard data augmentation and small batch sizes (128). We observe that among these methods only m-SAM with a low m (i.e., 128-SAM) substantially improves generalization.

In most of the past literature, sharpness is defined for $\mathcal{S} = \mathcal{S}_{train}$ (Keskar et al., 2016; Neyshabur et al., 2017; Jiang et al., 2019). However, Foret et al. (2021) recently introduced the notion of *m-sharpness* which is the average of the sharpness computed over all the batches $\mathcal{S}$ of size m from the training set $\mathcal{S}_{train}$.

Lower sharpness is correlated with lower test error (Keskar et al., 2016), however, the correlation is not always perfect (Neyshabur et al., 2017; Jiang et al., 2019). Moreover, the sharpness definition itself can be problematic since rescaling of incoming and outcoming weights of a node that leads to the same function can lead to very different sharpness values (Dinh et al., 2017). Kwon et al. (2021) suggest a sharpness definition that fixes this rescaling problem but other problems still exist such as the sensitivity of classification losses to the scale of the parameters (Neyshabur et al., 2017).

Sharpness-aware minimization. Foret et al. (2021) theoretically base the SAM algorithm on the following objective:

$$\textbf{\textit{n}-SAM:} \quad \min_{\boldsymbol{w} \in \mathbb{R}^{|w|}} \max_{\|\boldsymbol{\delta}\|_2 \leq \rho} \sum_{i=1}^{n} \ell_i(\boldsymbol{w} + \boldsymbol{\delta}), \tag{5.2}$$

which we denote as ***n*-SAM** since it is based on maximization of the sum of the losses over the n training points. They justify this objective via a PAC-Bayesian generalization bound, although they show empirically (see Fig. 3 therein) that the following objective leads to better generalization:

$$\textbf{\textit{m}-SAM:} \min_{\boldsymbol{w} \in \mathbb{R}^{|w|}} \sum_{\substack{\mathcal{S} \subset \mathcal{S}_{train}, \\ |\mathcal{S}|=m}} \max_{\|\boldsymbol{\delta}\|_2 \leq \rho} \sum_{i \in \mathcal{S}} \ell_i(\boldsymbol{w} + \boldsymbol{\delta}), \tag{5.3}$$

which we denote as ***m*-SAM** since it is based on maximization of the sum of the losses over batches of m training points and therefore related to the m-sharpness.

To make SAM practical, Foret et al. (2021) propose to minimize the m-SAM objective with stochastic gradients. Denoting the batch indices at time t by I_t ($|I_t| = m$), this leads to the following update rule on each iteration of training:

$$w_{t+1} = w_t - \frac{\gamma_t}{|I_t|} \sum_{i \in I_t} \nabla \ell_i \Big(w_t + \frac{\rho_t}{|I_t|} \sum_{j \in I_t} \nabla \ell_j(w_t) \Big). \tag{5.4}$$

Importantly, the *same* batch I_t is used for the inner and outer gradient steps. We note that ρ_t can optionally include the gradient normalization suggested in Foret et al. (2021), i.e., $\rho_t := \rho / \| \frac{1}{|I_t|} \sum_{j \in I_t} \nabla \ell_j(w_t) \|_2$. However, we show in Sec. 5.6 that its usage is not necessary for improving generalization, so we will omit it from our theoretical analysis.

Importance of low-m, worst-case perturbations. In order to improve upon ERM, Foret et al. (2021) use SAM with low-m and worst-case perturbations. To clearly illustrate the importance of these two choices, we show the performance of the following weight perturbation methods: no perturbations (ERM), random perturbations (prior to taking the gradient on each iteration), n-SAM, and 128-SAM. We use ResNet-18 on CIFAR-10 and ResNet-34 on CIFAR-100 (Krizhevsky and Hinton, 2009) with standard data augmentation and batch size 128 and refer to App. 5.11 for full experimental details, including our implementation of n-SAM. Fig. 5.1 clearly suggests that (1) the improvement from random perturbations is marginal, and (2) the only method that substantially improves generalization is low-m SAM (i.e., 128-SAM). Thus, worst-case perturbations and the use of m-sharpness in SAM are essential for the generalization improvement (which depends continuously on m as noted by Foret et al. (2021), see Fig. 5.16 in App. 5.12.1). We also note that using too low m is inefficient in practice since it does not fully utilize the computational accelerators such as GPUs. Thus, using higher m values (such as 128) helps to balance the generalization improvement with the computational efficiency. Finally, we note that using SAM with large batch sizes without using a smaller m leads to suboptimal generalization (see Fig. 5.17 in App. 5.12.2).

5.4 Challenging the Existing Understanding of SAM

In this section, we show the limitations of the current understanding of SAM. In particular, we discuss that the generalization bounds on which its only *formal* justification relies on (such as those presented in Foret et al. (2021); Wu et al. (2020b); Kwon et al. (2021)) cannot explain its success. Second, we argue that contrary to a common belief, convergence of SAM to flatter minima measured in terms of m-sharpness does not always translate to better generalization.

The existing generalization bound does not explain the success of SAM. The main theoretical justification for SAM comes from the PAC-Bayesian generalization bound presented, e.g., in Theorem 2 of Foret et al. (2021). However, the bound is derived for *random* perturbations of the parameters, i.e. the leading term of the bound is equal to $\mathbb{E}_{\delta \sim \mathcal{N}(0,\sigma)} \sum_{i=1}^n \ell_i(w+\delta)$. The extension to *worst-case* perturbations $\max_{\|\delta\|_2 \leq \rho} \sum_{i=1}^n \ell_i(w+$

δ), is done post hoc and only makes the bound less tight. Moreover, we can see empirically (Fig. 5.1) that *both* training methods suggested by the derivation of this bound (random perturbations and n-SAM) do not substantially improve generalization. This generalization bound can be similarly extended to m-SAM by upper bounding the leading term via the maximum taken over mini-batches. However, this bound would incorrectly suggest that 128-SAM should have the worst generalization among all the three weight-perturbation methods while it is the only method that successfully improves generalization.

We note that coming up with tight generalization bounds even for well-established ERM for overparametrized models is an open research question (Nagarajan and Kolter, 2019). One could expect, however, that at least the *relative* tightness of the bounds could reflect the correct ranking between the three methods, but it is not the case. Thus, we conclude that the existing generalization bound cannot explain the generalization improvement of low-m SAM.

A flatter minimum does not always lead to better generalization. One could assume that although the generalization bound that relies on m-sharpness is loose, m-sharpness can still be an important quantity for generalization. This is suggested by its better correlation with the test error compared to the sharpness computed on the whole training set (Foret et al., 2021). In particular, we could expect that convergence of SAM to better-generalizing minima can be explained by a lower m-sharpness of these minima. To check this hypothesis, we select multiple models trained with group normalization[1] that achieve zero training error and measure their m-sharpness for $m = 128$ and different perturbation radii ρ in Fig. 5.2. We note that the considered networks are not reparametrized in an adversarial way (Dinh et al., 2017) and they all use the same weight decay parameters which makes them more comparable to each other. First of all, we observe that *none* of the radii ρ gives the correct ranking between the methods according to their test error, although m-sharpness ranks correctly SAM and ERM for the same batch size. In particular, we see that the minimum found by SAM with a large batch size (1024) is flatter than the minimum found by ERM with a small batch size (128) although the ERM model leads to a better test error: 6.17% vs. 6.80% on CIFAR-10 and 25.06% vs. 28.31% on CIFAR-100. This shows that it is easy to find counterexamples where flatter minima generalize worse.

We further note that there are simple examples that illustrate that m-sharpness cannot be a universal quantity at distinguishing well-generalizing minima. E.g., consider a linear model $f_x(w) = \langle w, x \rangle$ and a decreasing margin-based loss ℓ, then the 1-sharpness has a closed-form solution:

$$\sum_{i=1}^{n} \max_{\|\boldsymbol{\delta}\|_2 \leq \rho} \ell\left(y_i \left\langle \boldsymbol{w} + \boldsymbol{\delta}, \boldsymbol{x}_i \right\rangle\right) - \ell\left(y_i \left\langle \boldsymbol{w}, \boldsymbol{x}_i \right\rangle\right) = \sum_{i=1}^{n} \ell\left(y_i \left\langle \boldsymbol{w}, \boldsymbol{x}_i \right\rangle - \rho \left\|\boldsymbol{x}_i\right\|_2\right) - \ell\left(y_i \left\langle \boldsymbol{w}, \boldsymbol{x}_i \right\rangle\right).$$

[1] We consider networks with group normalization (Wu and He, 2018) instead of the more common batch normalization (Ioffe and Szegedy, 2015) since we observed a large discrepancy between m-sharpness computed with the training-time vs. test-time batch normalization (see the experiment in Fig. 5.19 in App. 5.12.4).

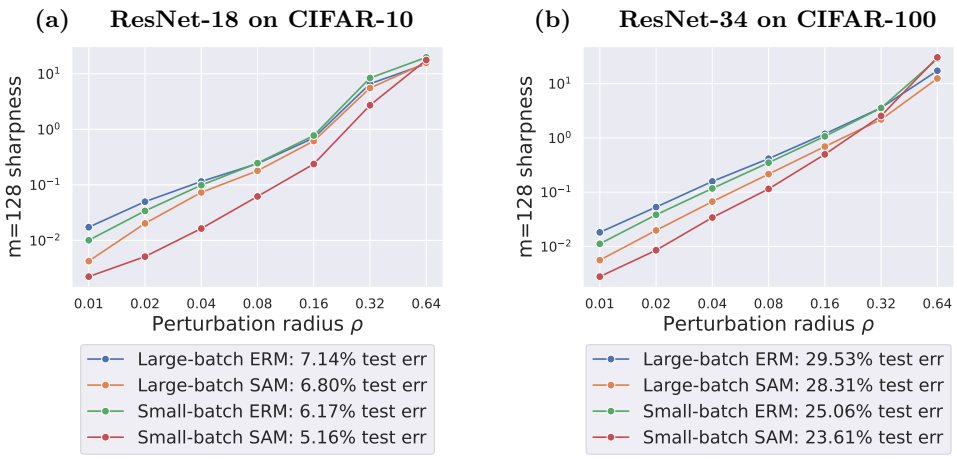


Figure 5.2: $m = 128$ sharpness computed over different perturbation radii ρ at the minima of ERM and SAM models trained with large (1024) and small batches (128). All models are trained with group normalization and achieve zero training error.

The 1-sharpness is influenced only by the term $-\rho \|\boldsymbol{x}_i\|_2$ which does not depend on a specific w. In particular, it implies that all global minimizers w^* of the training loss are *equally sharp* according to the 1-sharpness which, thus, cannot suggest which global minima generalize better.

Since (m-)sharpness does not always distinguish better- from worse-generalizing minima, the common intuition about sharp vs. flat minima (Keskar et al., 2016) can be incomplete. This suggests that it is likely that some other quantity is responsible for generalization which can be correlated with (m-)sharpness in *some* cases, but not always. This motivates us to develop a better understanding of the role of m in m-SAM, particularly on simpler models which are amenable for a theoretical study.

5.5 Understanding the Generalization Benefits of SAM

In this section, we first check empirically whether the advantage of lower m in m-SAM comes from a more accurate solution of the inner maximization problem or from specific properties of batch normalization. We conclude that it is not the case and hypothesize that the advantage comes rather from a better implicit bias of gradient descent induced by m-SAM. We characterize this implicit bias for diagonal linear networks showing that SAM can *provably* improve generalization, and the improvement is larger for 1-SAM than for n-SAM. Then we complement the theoretical results with experiments on deep networks showing a few intriguing properties of SAM.

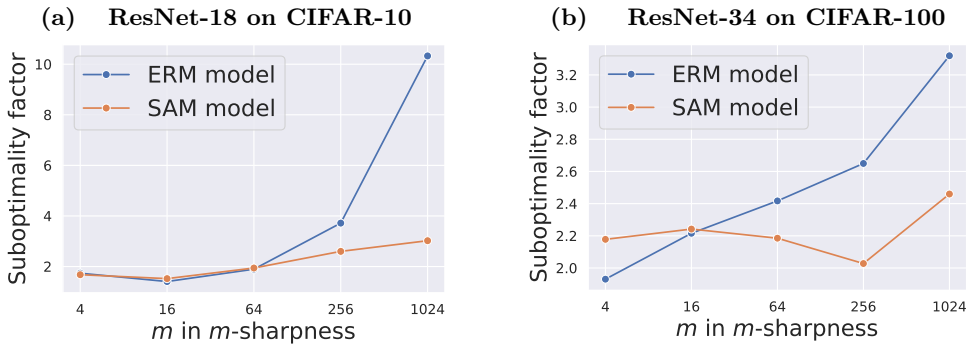


Figure 5.3: Suboptimality factor of m-sharpness ($\rho = 0.1$) computed using 100 steps of projected gradient ascent compared to only 1 step for ERM and SAM models with group normalization.

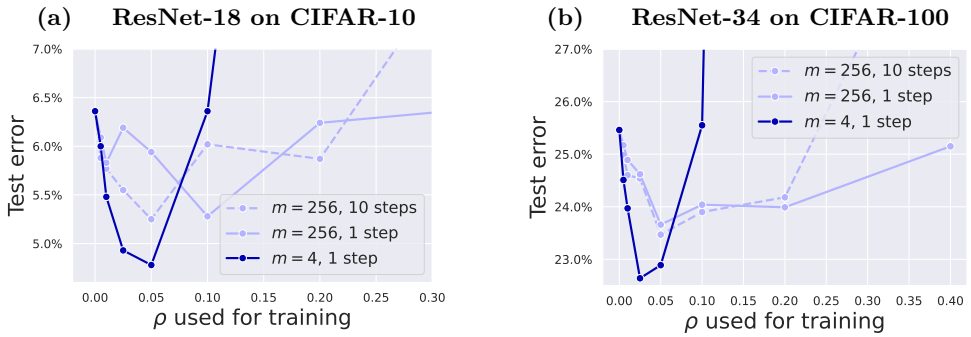


Figure 5.4: Test error of SAM models with group normalization trained with different numbers of projected gradient ascent steps (10 vs. 1) for m-SAM and different m values (256 vs. 4) using batch size 256.

5.5.1 Testing Two Natural Hypotheses for Why Low m in m-SAM Could be Beneficial

As illustrated in Fig. 5.1, the success of m-SAM fully relies on the effect of low m which is, however, remains unexplained in the current literature. As a starting point, we could consider the following two natural hypotheses for why low m could be beneficial.

Hypothesis 1: **lower m leads to more accurate maximization.** Since m-SAM relies only on a *single* step of projected gradient ascent for the inner maximization problem in Eq. (5.3), it is unclear in advance how accurately this problem is solved. One could assume that using a lower m can make the single-step solution more accurate as intuitively the function which is being optimized might become "simpler" due to fewer terms in the summation. Indeed, there is evidence towards this hypothesis: Fig. 5.3 shows the suboptimality factor between m-sharpness computed using 100 steps vs. 1 step of projected gradient ascent for $\rho = 0.1$ (the optimal ρ for 256-SAM in terms of generalization) for ERM and SAM models. We can see that the suboptimality factor tends to increase over m and can be as large as $10\times$ for the ERM model on CIFAR-10 for $m = 1024$. This finding suggests that the standard single-step m-SAM can indeed fail to find an accurate

maximizer and the value of m can have a significant impact on it. However, despite this fact, using multiple steps in SAM *does not* improve generalization as we show in Fig. 5.4. E.g., on CIFAR-10 it merely leads to a shift of the optimal ρ from 0.1 to 0.05, without noticeable improvements of the test error. This is also in agreement with the observation from Foret et al. (2021) on why including second-order terms can slightly hurt generalization: solving the inner maximization problem more accurately leads to the fact that the same radius ρ can become effectively too large (as on CIFAR-10) leading to worse performance.

Hypothesis 2: lower m results in a better regularizing effect of batch normalization. As pointed out in Hoffer et al. (2017) and Goyal et al. (2017), batch normalization (BN) has a beneficial regularization effect that depends on the mini-batch size. In particular, using the BN statistics from a smaller subbatch is coined as *ghost batch normalization* (Hoffer et al., 2017) and tends to improve generalization. Thus, it could be the case that the generalization improvement of m-SAM is due to this effect as its implementation assumes using a smaller subbatch of size m. To test this hypothesis, in Fig. 5.4, we show results of networks trained instead with *group normalization* that does not lead to any extra dependency on the effective batch size. We can see that a significant generalization improvement by m-SAM is still achieved for low m ($m = 4$ for batch size 256), and this holds for both datasets. Thus, the generalization improvement of m-SAM is not specific to BN.

We hypothesize instead that low-m SAM leads to a better *implicit* bias of gradient descent for commonly used neural network architectures, meaning that some important complexity measure of the model gets implicitly minimized over training that may not be obviously linked to m-sharpness.

5.5.2 Provable Benefit of SAM for Diagonal Linear Networks

Here we theoretically study the implicit bias of full-batch 1-SAM and n-SAM for diagonal linear networks on a sparse regression problem. We show that 1-SAM has a better implicit bias than ERM *and* n-SAM which explains its improved generalization in this setting.

Implicit bias of 1-SAM and n-SAM. The implicit bias of gradient methods is well understood for overparametrized linear models where all gradient-based algorithms enjoy the same implicit bias towards minimization of the ℓ_2-norm of the parameters. For diagonal linear neural networks, where a linear predictor $\langle \beta, x \rangle$ can be parametrized via $\beta = w_+^2 - w_-^2$ [2] with a parameter vector $w = \left[\begin{smallmatrix} w_+ \\ w_- \end{smallmatrix}\right] \in \mathbb{R}^{2d}$, first-order algorithms have a richer implicit bias. We consider here an overparametrized sparse regression problem, meaning that the ground truth β^* is a sparse vector, with the squared loss:

[2] See Woodworth et al. (2020) for why this parametrization is equivalent to a diagonal network $\beta = u \odot v$. Moreover, the signs of u_i and v_i will not change throughout training, hence the use of the notation w_+ and w_-.

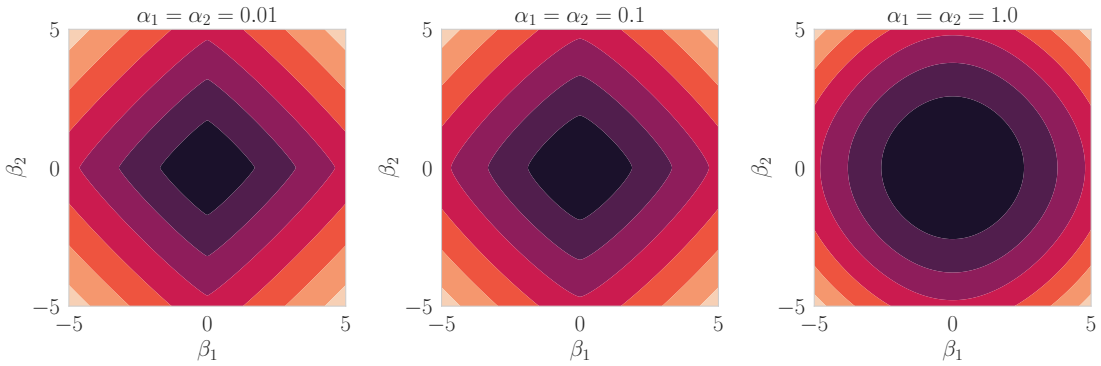


Figure 5.5: Illustration of the hyperbolic entropy $\phi_\alpha(\beta)$ for $\beta \in \mathbb{R}^2$ that interpolates between $\|\beta\|_1$ for small α and $\|\beta\|_2$ for large α.

$$L(w) := \frac{1}{4n} \sum_{i=1}^{n} (\langle w_+^2 - w_-^2, x_i \rangle - y_i)^2, \tag{5.5}$$

where overparametrization means that $n \ll d$ and there exist many w such that $L(w) = 0$. We note that in our setting, any global minimizer w^* of $L(w^*)$ is also a global minimizer for the m-SAM algorithm for any $m \in \{1, \dots, n\}$ since all per-example gradients are zero and hence the ascent step of SAM will not modify w^*. Thus, any difference in generalization between m-SAM and ERM has to be attributed rather to the *implicit* bias of each of these algorithms.

We first recall the seminal result of Woodworth et al. (2020) and refer the readers to App. 5.9 for further details. Assuming global convergence, the solution selected by the gradient flow initialized as $w_+ = w_- = \alpha \in \mathbb{R}^d_{>0}$ and denoted β_∞^α solves the following constrained optimization problem:

$$\beta_\infty^\alpha = \underset{\beta \in \mathbb{R}^d \text{ s.t. } X\beta = y}{\arg\min} \phi_\alpha(\beta), \tag{5.6}$$

where the potential ϕ_α is given as $\phi_\alpha(\beta) = \sum_{i=1}^d \alpha_i^2 q(\beta_i / \alpha_i^2)$ with $q(z) = 2 - \sqrt{4 + z^2} + z \operatorname{arcsinh}(z/2)$. As illustrated in Fig. 5.5, ϕ_α interpolates between the ℓ_1 and the ℓ_2 norms of β according to the initialization scale α. Large α's lead to low ℓ_2-type solutions, while small α's lead to low ℓ_1-type solutions which are known to induce good generalization properties for sparse problems (Woodworth et al., 2020).

Our main theoretical result is that both 1-SAM and n-SAM dynamics, when considered in their full-batch version (see Sec. 5.8 for details), bias the flow towards solutions which minimize the potential ϕ_α but with effective parameters $\alpha_{\text{1-SAM}}$ and $\alpha_{\text{n-SAM}}$ which are strictly smaller than α for a suitable inner step size ρ. In addition, typically $\|\alpha_{\text{1-SAM}}\|_1 < \|\alpha_{\text{n-SAM}}\|_1$ and, therefore, the solution chosen by 1-SAM has better sparsity-inducing properties than the solution of n-SAM and standard ERM.

Theorem 5.5.1 (Informal)**.** *Assuming global convergence, the solutions selected by the full-batch versions of the* 1*-SAM and* n*-SAM algorithms taken with infinitesimally small*

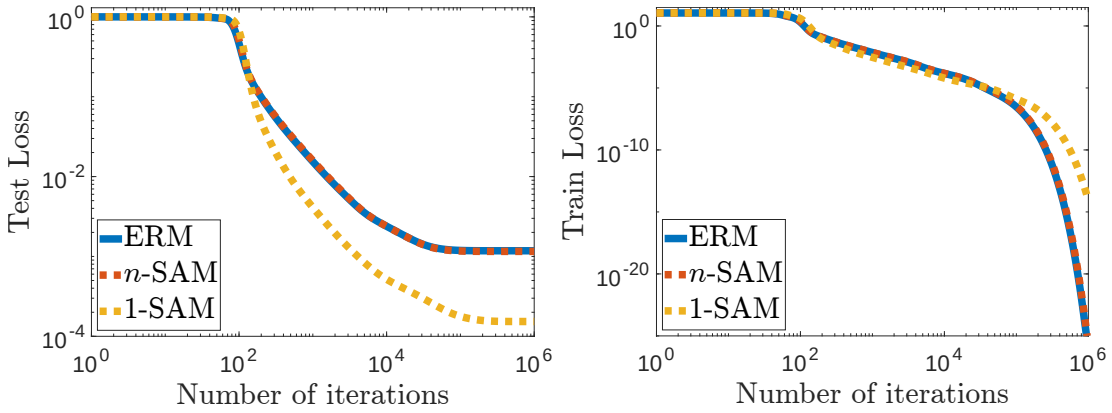


Figure 5.6: Implicit bias of 1-SAM and n-SAM compared to ERM for a diagonal linear network on a sparse regression problem. We can see that 1-SAM generalizes significantly better than n-SAM and ERM.

step sizes and initialized at $w_+ = w_- = \alpha \in \mathbb{R}^d_{>0}$*, solve the optimization problem (5.6) with effective parameters:*

$$\alpha_{1\text{-}SAM} = \alpha \odot e^{-\rho \Delta_{1\text{-}SAM} + O(\rho^2)}, \quad \alpha_{n\text{-}SAM} = \alpha \odot e^{-\rho \Delta_{n\text{-}SAM} + O(\rho^2)},$$

where $\Delta_{1\text{-}SAM}, \Delta_{n\text{-}SAM} \in \mathbb{R}^d_+$ *for which typically:*

$$\|\Delta_{1\text{-}SAM}\|_1 \approx d \int_0^\infty L(w(s))ds \quad \text{and}$$
$$\|\Delta_{n\text{-}SAM}\|_1 \approx \frac{d}{n} \int_0^\infty L(w(s))ds.$$

The results are formally stated in Theorem 5.9.2 and 5.9.3 in App. 5.9. 1-SAM has better implicit bias properties since its effective scale of α is considerably smaller than the one of n-SAM due to the lack of the $\frac{1}{n}$ factor in the exponent. It is worth noting that the vectors $\Delta_{\text{1-SAM}}$ and $\Delta_{\text{n-SAM}}$ are linked with the integral of the loss function along the flow. Thereby, the speed of convergence of the training loss impacts the magnitude of the biasing effect: the slower the convergence, the better the bias, similarly to what is observed for SGD in Pesme et al. (2021). Extending this result to stochastic implementations of 1-SAM and n-SAM algorithms could be done following Pesme et al. (2021) but is outside of the scope of this paper.

Empirical evidence for the implicit bias. We compare the training and test loss of ERM, 1-SAM, and n-SAM in Fig. 5.6 for the same perturbation radius ρ, and for different ρ in App. 5.9.3 (Fig. 5.14). As predicted, the methods show different generalization abilities: ERM and n-SAM achieve approximately the same performance whereas 1-SAM clearly benefits from a better implicit bias. This is coherent with the deep learning experiments presented in Fig. 5.1 on CIFAR-10 and CIFAR-100. We also note that the training loss of all the variants is converging to zero but the convergence of 1-SAM is slower. Additionally, we show a similar experiment with *stochastic* variants of the algorithms in App. 5.9.3

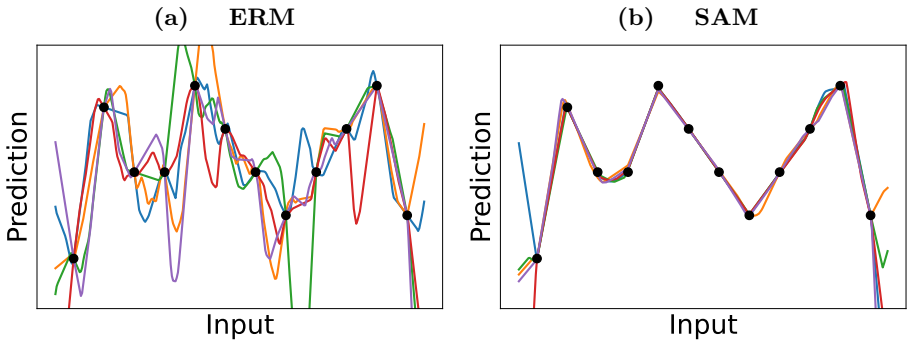


Figure 5.7: The effect of the implicit bias of ERM vs. SAM for a one hidden layer ReLU network trained with full-batch gradient descent. Each run is replicated over five random initializations.

(Fig. 5.13) where their performance is, as expected, better compared to their deterministic counterparts.

5.5.3 Empirical Study of the Implicit Bias in Non-Linear Networks

Here we conduct a series of experiments to characterize the implicit bias of SAM on *non-linear* networks.

The sparsity-inducing bias of SAM for a simple ReLU network. We start from the simplest non-linear network: a one hidden layer ReLU network applied to a simple 1D regression problem from Blanc et al. (2020). We use it to illustrate the implicit bias of SAM in terms of the geometry of the learned function. For this, we train ReLU networks with 100 hidden units using full-batch gradient descent on the quadratic loss with ERM and SAM[3] over five different random initializations. We plot the resulting functions in Fig. 5.7. We observe that SAM leads to simpler interpolations of the data points than ERM, and it is much more stable over random initializations. In particular, SAM seems to be biased toward a sparse combination of ReLUs which is reminiscent of Chizat and Bach (2020b) who show that the limits of the gradient flow can be described as a max-margin classifier that favors hidden low-dimensional structures by implicitly regularizing the $\mathcal{F}_1$ variation norm. Moreover, this also relates to our Theorem 5.5.1 where sparsity rather shows up in terms of the lower ℓ_1-norm of the resulting linear predictor. This further illustrates that there can exist multiple ways in which one can describe the beneficial effect of SAM. For deep non-linear networks, however, the effect of SAM is hard to visualize, but we can still characterize some of its important properties.

The effect of SAM for deep networks at different stages of training. To develop a better understanding of the implicit bias of SAM for deep networks, we can analyze at which stages of training using SAM is necessary to get generalization benefits. One could assume, for example, that its effect is important *only* early in training so that the first updates of SAM steer the optimization trajectory towards a better-generalizing minimum.

[3]Since $n = 12$ for this task, we observed no substantial difference between 1-SAM and n-SAM.

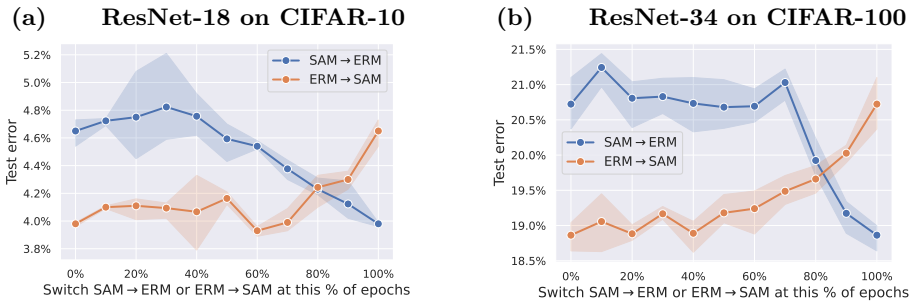


Figure 5.8: Test error of SAM → ERM and ERM → SAM when the methods are switched at different % of epochs. For example, for SAM → ERM, 0% corresponds to ERM and 100% corresponds to SAM. We observe that a method which is run at the beginning of training has little influence on the final performance.

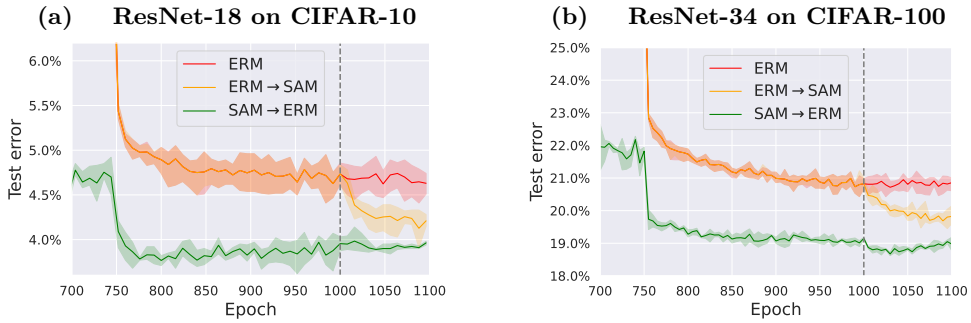


Figure 5.9: Test error over epochs for ERM compared to ERM → SAM and SAM → ERM training where the methods are switched only at the end of training. In particular, we can see that SAM can gradually escape the worse-generalizing minimum found by ERM.

In that case, switching from SAM to ERM would not degrade the performance. To better understand this, we train models first with SAM and then switch to ERM for the remaining epochs (SAM → ERM) and also do a complementary experiment by switching from ERM to SAM (ERM → SAM) and show results in Fig. 5.8. Interestingly, we observe that a method that is used at the beginning of training has little influence on the final performance. E.g., when SAM is switched to ERM within the first 70% epochs on CIFAR-100, the resulting model generalizes as well as ERM. Furthermore, we note a high degree of continuity of the test error with respect to the number of epochs at which we switch the methods. This does not support the idea that the models converge to some entirely distinct minima and instead suggests convergence to different minima in a connected valley where some directions generalize progressively better. Another intriguing observation is that enabling SAM only towards the end of training is sufficient to get a significant improvement in terms of generalization. We discuss this phenomenon next in more detail.

The importance of the implicit bias of SAM at the end of training. We take a closer look on the performance of ERM → SAM and SAM → ERM when we switch between the methods only for the last ≈ 10% of epochs in Fig. 5.9 where we plot the test error over epochs. First, we see that for SAM → ERM, once SAM converges to a

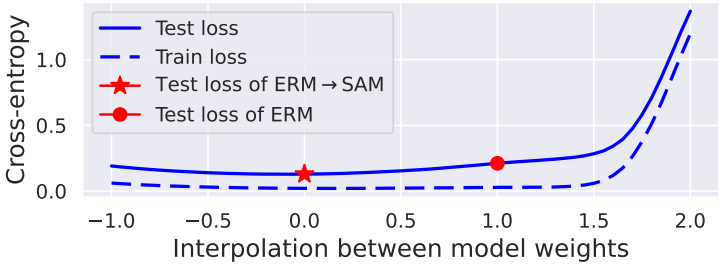


Figure 5.10: Loss interpolations between $w_{ERM \to SAM}$ and w_{ERM} for a ResNet-18 trained on CIFAR-10.

well-generalizing minimum thanks to its implicit bias, then it is not important whether we continue optimization with SAM or with ERM, and we do not observe significant overfitting when switching to ERM. At the same time, for ERM $\to$ SAM we observe a different behavior: the test error clearly improves when switching from ERM to SAM. This suggests that SAM (using a higher ρ than the standard value, see App. 5.11) can gradually escape the worse-generalizing minimum which ERM converged to. This phenomenon is interesting since it suggests a *practically relevant* fine-tuning scheme that can save computations as we can start from any pre-trained model and substantially improve its generalization. Moreover, interestingly, the final point of the ERM $\to$ SAM model is situated *in the same basin* as the original ERM model as we show in Fig. 5.10 which resembles the asymmetric loss interpolations observed previously for stochastic weight averaging (He et al., 2019).

We make very similar observations regarding fine-tuning with SAM and linear connectivity also on a diagonal linear network as shown in App. 5.9.3 (Fig. 5.15). We believe the observations from Fig. 5.9 can be explained by our Theorem 5.5.1 which shows that for diagonal linear networks, the key quantity determining the magnitude of the implicit bias for SAM is the integral of the loss over the optimization trajectory $w(s)$. In the case of ERM $\to$ SAM, the integral is taken only over the last epochs but this can still be sufficient to improve the biasing effect. At the same time, for SAM $\to$ ERM, the integral is already large enough due to the first 1000 epochs with SAM and switching back to ERM preserves the implicit bias. We discuss it in more detail in App. 5.9.3.

5.6 Understanding the Optimization Aspects of SAM

The results on the implicit bias of SAM presented above require that the algorithm converges to zero training error. In the current literature, however, a convergence analysis (even to a stationary point) is missing for SAM. In particular, we do not know what are the conditions on the training ERM loss, inner step size γ_t, and perturbation radius ρ_t so that SAM is guaranteed to converge. We also do not know whether SAM converges to a stationary point of the ERM objective. To fill in this gap, we first theoretically study convergence of SAM and then relate the theoretical findings with empirical observations

on deep networks.

5.6.1 Theoretical Analysis of Convergence of SAM

Here we show that SAM leads to convergence guarantees in terms of the standard training loss. In the following, we analyze the convergence of the m-SAM algorithm whose update rule is defined in Eq. (5.4). We make the following assumptions on the training loss $L(w) = \frac{1}{n}\sum_{i=1}^{n} \ell_i(w)$:

(A1) (Bounded variance). *There exists* $\sigma \geq 0$ *s.t.* $\mathbb{E}[\|\nabla \ell_i(w) - \nabla L(w)\|^2] \leq \sigma^2$ *for all* $i \sim \mathcal{U}(\llbracket 1, n \rrbracket)$ *and* $w \in \mathbb{R}^d$.

(A2) (Individual β-smoothness). *There exists* $\beta \geq 0$ *s.t.* $\|\nabla \ell_i(w) - \nabla \ell_i(v)\| \leq \beta \|w - v\|$ *for all* $w, v \in \mathbb{R}^d$ *and* $i \in \llbracket 1, n \rrbracket$.

(A3) (Polyak-Lojasiewicz). *There exists* $\mu > 0$ *s.t.* $\frac{1}{2}\|\nabla L(w)\|^2 \geq \mu(L(w) - L_*)$ *for all* $w, v \in \mathbb{R}^d$.

Both assumptions **(A1)** and **(A2)** are standard in the optimization literature and should hold for neural networks with smooth activations and losses (such as cross-entropy). The assumption **(A2)** requires the inputs to be bounded but this is typically satisfied (e.g., images are all in $[0, 1]^d$). The assumption **(A3)** corresponds to easier problems (e.g., strongly convex ones) for which global convergence can be proven. We have the following convergence result:

Theorem 5.6.1. *Assume* ***(A1)*** *and* ***(A2)*** *for the iterates (5.4). Then for any number of iterations* $T \geq 0$*, batch size* b*, and step sizes* $\gamma_t = \frac{1}{\sqrt{T}\beta}$ *and* $\rho_t = \frac{1}{T^{1/4}\beta}$*, we have:*

$$\frac{1}{T}\mathbb{E}\left[\sum_{t=0}^{T-1} \|\nabla L(w_t)\|^2\right] \leq \frac{4\beta}{\sqrt{T}}(L(w_0) - L_*) + \frac{8\sigma^2}{b\sqrt{T}},$$

In addition, under ***(A3)****, with step sizes* $\gamma_t = \min\{\frac{8t+4}{3\mu(t+1)^2}, \frac{1}{2\beta}\}$ *and* $\rho_t = \sqrt{\gamma_t/\beta}$*:*

$$\mathbb{E}\left[L(w_T)\right] - L_* \leq \frac{3\beta^2(L(w_0) - L_*)}{\mu^2 T^2} + \frac{22\beta\sigma^2}{\mu^2 bT}.$$

We provide the proof in App. 5.10.2 and make several remarks:

- We recover the rates of SGD with the usual condition on the step size γ_t (Ghadimi and Lan, 2013; Karimi et al., 2016).
- The ascent step size ρ_t, however, has to be $O(\sqrt{\gamma_t})$ to ensure convergence, i.e., it tolerates a slower decrease than γ_t. This finding is aligned with the observation that the ascent step size should not be decreased as drastically as the descent step size when training neural networks (see Fig. 5.21 in App. 5.12.6).

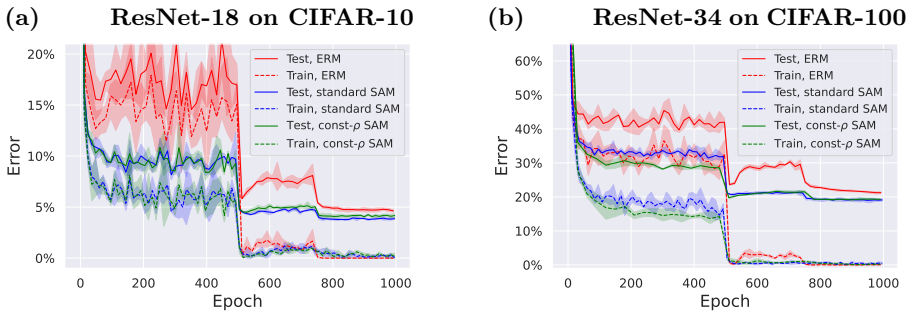


Figure 5.11: Training and test error of ERM, standard SAM, and SAM with a constant step size ρ (i.e., without gradient normalization) over epochs. We can see that both ERM and SAM converge to zero training error and the gradient normalization is not crucial for SAM.

- On the technical side, the proof relies on the bound $\langle \nabla L(w_t + \eta \nabla L(w_t)), \nabla L(w_t) \rangle \geq (1 - \eta\beta)\|\nabla L(w_t)\|^2$ which shows that SAM-step is well aligned with the gradient step (see Lemma 7 in App. 5.10.2).

5.6.2 Convergence of SAM for Deep Networks

Here we relate the convergence analysis to empirical observations for deep learning tasks.

Both ERM and SAM converge for deep networks. We compare the behavior of ERM and SAM by training a ResNet-18 on CIFAR-10 and CIFAR-100 for 1000 epochs (see App. 5.11 for experimental details) and plot the results over epochs in Fig. 5.11. We observe that not only the ERM model but also the model trained with SAM fits all the training points and converges to a *nearly zero training loss*: 0.0013 ± 0.00002 for ERM vs 0.0034 ± 0.0004 for SAM on CIFAR-10. However, the SAM model has significantly better generalization performance due to its implicit bias: $4.75\% \pm 0.14\%$ vs. $3.94\% \pm 0.09\%$ test error. Moreover, we observe no noticeable overfitting throughout training: the best and last model differ by at most 0.1% test error for both methods. Finally, we note that the behavior of ERM vs. SAM on CIFAR-100 is qualitatively similar.

Performance of SAM with constant step sizes ρ_t. Our convergence proof in Sec. 5.6.1 for non-convex objectives relies on constant step sizes ρ_t. However, the standard SAM algorithm as introduced in Foret et al. (2021) uses step sizes ρ_t inversely proportional to the gradient norm. Thus, one can wonder if such step sizes are important for achieving better convergence or generalization. Fig. 5.11 shows that on CIFAR-10 and CIFAR-100, both methods converge to zero training error at a similar speed. Moreover, they achieve similar improvements in terms of generalization: $3.94\% \pm 0.09\%$ test error for standard SAM vs. $4.15\% \pm 0.16\%$ for SAM with constant ρ_t on CIFAR-10. For CIFAR-100, the test error matches almost exactly: $19.22\% \pm 0.38\%$ vs. $19.30\% \pm 0.38\%$. We also note that the optimal ρ differs for both formulations: $\rho_t = 0.2/\|\nabla\|_2$ with normalization vs. $\rho_t = 0.3$ without normalization, so simply removing the gradient normalization without doing a new grid search over ρ_t can lead to suboptimal results.

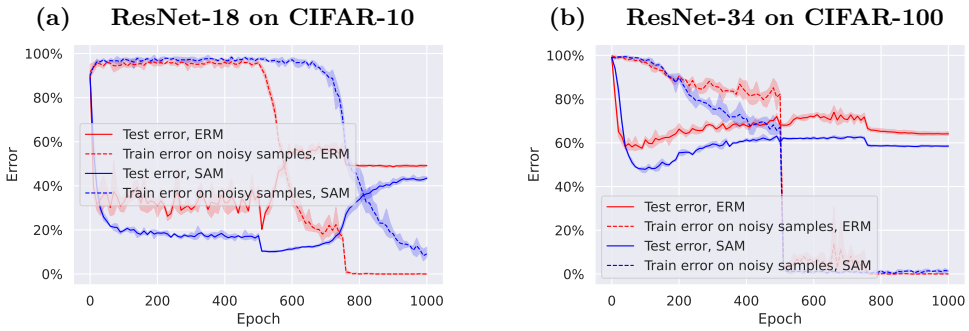


Figure 5.12: Error rates of ERM and SAM over epochs on CIFAR-10 and CIFAR-100 *with 60% label noise*. We see that the test error increases when the models fit the noisy samples.

Is it always beneficial for SAM to converge to zero loss? Here we consider the setting of uniform label noise, i.e., when a fraction of the training labels is changed to random labels and kept fixed throughout the training. This setting differs from the standard noiseless case (typical for many vision datasets such as CIFAR-10) as converging to nearly zero training loss is harmful for ERM and leads to substantial overfitting. Thus, one could assume that the beneficial effect of SAM in this setting can come from preventing convergence and avoiding fitting the label noise. We plot test error and training error on noisy samples for a ResNet-18 trained on CIFAR-10 and CIFAR-100 with 60% label noise in Fig. 5.12. We see that SAM noticeably improves generalization over ERM, although later in training SAM also starts to fit the noisy points which is in agreement with the convergence analysis. In App. 5.12.7, we confirm the same findings for SAM with constant ρ_t. Thus, SAM also requires early stopping either explicitly via a validation set or implicitly via restricting the number of training epochs as done, e.g., in Foret et al. (2021). Interestingly, this experiment also suggests that the beneficial effect of SAM is observed not only close to a minimum but also along the whole optimization trajectory. Overall, we conclude that SAM can easily overfit and its convergence in terms of the training loss can be a negative feature for datasets with noisy labels.

5.7 Conclusions

We showed why the existing justifications for the success of m-SAM based on generalization bounds and the idea of convergence to flat minima are incomplete. We hypothesized that there exists some other quantity which is responsible for the improved generalization of m-SAM which is implicitly minimized. We analyzed the implicit bias of 1-SAM and n-SAM for diagonal linear networks showing that the implicit quantity which is minimized is related to the ℓ_1-norm of the resulting linear predictor, and it is stronger for 1-SAM than for n-SAM. We further studied the properties of the implicit bias on non-linear networks empirically where we showed that fine-tuning an ERM model with SAM can lead to significant generalization improvements. Finally, we provided convergence results of SAM for non-convex objectives when used with stochastic gradient which we confirmed empirically for deep networks and discussed its relation to the generalization behavior of

SAM.

Appendix

5.8 Implementations of the SAM Algorithm in the Full-Batch Setting

We define here the implementations of the m-SAM algorithm in the *full-batch* setting for the two extreme values of m we consider, i.e., $m = 1$ and $m = n$. They correspond to the following objectives:

$$n\textbf{-SAM}: \min_{\boldsymbol{w}\in\mathbb{R}^{|w|}} \max_{\|\boldsymbol{\delta}\|_2\leq\rho} \frac{1}{n}\sum_{i=1}^{n} \ell_i(\boldsymbol{w}+\boldsymbol{\delta}), \qquad 1\textbf{-SAM}: \min_{\boldsymbol{w}\in\mathbb{R}^{|w|}} \frac{1}{n}\sum_{i=1}^{n} \max_{\|\boldsymbol{\delta}\|_2\leq\rho} \ell_i(\boldsymbol{w}+\boldsymbol{\delta}). \quad (5.7)$$

The update rule of the SAM algorithm for these objectives amounts to a variant of gradient descent with step size γ_t where the gradients are taken at intermediate points $w^i_{t+1/2}$, i.e., $w_{t+1} = w_t - \frac{\gamma_t}{n}\sum_{i=1}^{n} \nabla\ell_i(w^i_{t+1/2})$. The updates, however, differ in how the points $w^i_{t+1/2}$ are computed since they approximately maximize different functions with inner step sizes ρ_t:

$$n\textbf{-SAM:}\ \ w^i_{t+1/2} = w_t + \frac{\rho_t}{n}\sum_{j=1}^{n} \nabla\ell_j(w_t), \qquad 1\textbf{-SAM:}\ \ w^i_{t+1/2} = w_t + \rho_t\nabla\ell_i(w_t). \quad (5.8)$$

To make the SAM algorithm practical, Foret et al. (2021) propose to combine SAM with stochastic gradients which corresponds to the m-SAM algorithm defined in Eq. (5.4) in the main part.

5.9 Theoretical Analysis of the Implicit Bias for Diagonal Linear Networks

To understand why m-SAM is generalizing better than ERM, we consider the simpler problem of noiseless regression with 2-layer diagonal linear network for which we can precisely characterize the implicit bias of different optimization algorithms.

Optimization algorithms. We consider minimizing the training loss $L(w)$ using the following optimization algorithms:

- Gradient descent with an infinitesimally small step size, i.e., the gradient flow limit:

$$\dot{w}_t = -\nabla L(w_t). \quad (5.9)$$

- The n-SAM algorithm from Eq. (5.8) taken with an infinitesimally small outer step size and inner step size $\rho \geq 0$:

$$\dot{w}_t = -\nabla L(w_t + \rho \nabla L(w_t)). \tag{5.10}$$

- The 1-SAM algorithm from Eq. (5.8) taken with an infinitesimally small outer step size and inner step size $\rho \geq 0$:

$$\dot{w}_t = -\frac{1}{n}\sum_{i=1}^{n} \nabla \ell_i(w_t + \rho \nabla \ell_i(w_t)). \tag{5.11}$$

Previous work: implicit bias of the gradient flow. We first define the function ϕ_α for $\alpha \in \mathbb{R}^d$ which will be very useful to precisely characterize the implicit bias of the optimization algorithms we consider:

$$\phi_\alpha(\beta) = \sum_{i=1}^{d} \alpha_i^2 q(\beta_i/\alpha_i^2) \text{ where } q(z) = \int_0^z \operatorname{arcsinh}(u/2)du = 2 - \sqrt{4+z^2} + z\operatorname{arcsinh}(z/2). \tag{5.12}$$

Following Woodworth et al. (2020), one can show the following result for the gradient flow dynamics in Eq. (5.9).

Theorem 5.9.1 (Theorem 1 of Woodworth et al. (2020))**.** *If the solution β_∞ of the gradient flow (5.9) started from $w_+ = w_- = \alpha \in \mathbb{R}^d_{>0}$ for the squared parameter problem in Eq. (5.5) satisfies $X\beta_\infty = y$, then*

$$\beta_\infty = \arg\min_{\beta \in \mathbb{R}^d} \phi_\alpha(\beta) \quad s.t. \quad X\beta = y, \tag{5.13}$$

where ϕ_α is defined in Eq. (5.12).

It is worth noting that the implicit regularizer ϕ_α interpolates between the ℓ_1 and ℓ_2 norms (see Woodworth et al., 2020, Theorem 2). Therefore the scale of the initialization determines the implicit bias of the gradient flow. The algorithm, started from α, converges to the minimum ℓ_1-norm interpolator for small α and to the minimum ℓ_2-norm interpolator for large α. The proof follows from (a) the KKT condition for the optimization problem (5.13): $\nabla\phi_\alpha(w) = X^\top \nu$ for a Lagrange multiplier ν and (b) the closed form solution obtained by integrating the gradient flow, $w = b(X^\top \nu)$ for some function b and some vector ν. Identifying $\nabla\phi_\alpha(w) = b^{-1}(w)$ leads to the solution. Considering the same proof technique, we now derive the implicit bias for the n-SAM and 1-SAM algorithms.

5.9.1 Implicit Bias of the n-SAM Algorithm.

We start from characterizing the implicit bias of the n-SAM dynamics (5.10) in the following theorem using the function ϕ_α defined in Eq. (5.12). We will also make use of this notation: a parameter vector $w = \left[\begin{smallmatrix} w_+ \\ w_- \end{smallmatrix}\right] \in \mathbb{R}^{2d}$, a concatenation of matrices $\tilde{X} = [X \quad -X] \in \mathbb{R}^{n \times 2d}$ and a residual vector $r(t) = \tilde{X}w(t)^2 - y$.

Theorem 5.9.2. *If the solution β_∞ of the n-SAM gradient flow (5.10) started from $w_+ = w_- = \alpha \in \mathbb{R}^d_{>0}$ for the squared parameter problem in Eq. (5.5) satisfies $X\beta_\infty = y$, then*

$$\beta_\infty = \arg\min_\beta \phi_{\alpha_{n\text{-}SAM}}(\beta) \quad s.t. \quad X\beta = y,$$

where $\alpha_{n\text{-}SAM} = \alpha \odot \exp\left(-\frac{2\rho}{n^2}\int_0^\infty (X^\top r_s)^2 ds + O(\rho^2)\right)$.

We note that for a small enough ρ, the implicit bias parameter $\alpha_{\text{n-SAM}}$ is smaller than α. The scale of the vector $\frac{1}{n^2}\int_0^\infty (X^\top r_s)^2 ds$ which influences the implicit bias effect is related to the loss integral $\frac{d}{n}\int_0^\infty L(w(s))ds$ since $\|r_s\|^2 = nL(w(s))$ (see intuition in Eq. (5.19)). Thereby the speed of convergence of the loss controls the magnitude of the biasing effect. However in the case of n-SAM, as explained in Sec. 5.9.3, this effect is typically negligible because of the extra prefactor $\frac{d}{n}$ and this implementation behaves similarly as ERM as shown in the experiments in Sec. 5.5.2.

Proof. We follow the proof technique of Woodworth et al. (2020). We denote the intermediate step of n-SAM as $w_{sam}(t) = w(t) + \rho\nabla L(w(t))$ and the residual of $w_{sam}(t)$ as $r_{sam}(t) = \tilde{X} w_{sam}(t)^2 - y$. We start from deriving the equation satisfied by the flow

$$\begin{aligned}
\dot{w}(t) &= -\nabla L(w_{sam}(t)) \\
&= -\frac{1}{n}\tilde{X}^\top r_{sam}(t) \odot w_{sam}(t) \\
&= -\frac{1}{n}\tilde{X}^\top r_{sam}(t) \odot \left(w(t) + \frac{\rho}{n}\left(\tilde{X}^\top r(t)\right) \odot w(t)\right).
\end{aligned}$$

Now we can directly integrate this ODE to obtain an expression for $w(t)$:

$$w(t) = w(0) \odot \exp\left(-\frac{1}{n}\tilde{X}^\top \int_0^t r_{sam}(s)ds\right) \odot \exp\left(-\frac{\rho}{n^2}\int_0^t \left(\tilde{X}^\top r_{sam}(s)\right) \odot \left(\tilde{X}^\top r(s)\right) ds\right).$$

Using that the flow is initialized at $w(0) = \alpha$ and the definition of $\beta(t)$ yields to

$$\begin{aligned}
\beta(t) &= w_+(t)^2 - w_-(t)^2 \\
&= \alpha^2 \odot \exp\left(-\frac{2}{n}X^\top \int_0^t r_{sam}(s)ds\right) \odot \exp\left(-\frac{2\rho}{n^2}\int_0^t \left(X^\top r_{sam}(s)\right) \odot \left(X^\top r(s)\right) ds\right) \\
&\quad - \alpha^2 \odot \exp\left(\frac{2}{n}X^\top \int_0^t r_{sam}(s)ds\right) \odot \exp\left(-\frac{2\rho}{n^2}\int_0^t \left(X^\top r_{sam}(s)\right) \odot \left(X^\top r(s)\right) ds\right) \\
&= 2\alpha^2 \odot \exp\left(-\frac{2\rho}{n^2}\int_0^t \left(X^\top r_{sam}(s)\right) \odot \left(X^\top r(s)\right) ds\right) \odot \sinh\left(-\frac{2}{n}X^\top \int_0^t r_{sam}(s)ds\right).
\end{aligned}$$

Recall that we are assuming that β_∞ is a global minimum of the loss, i.e., $X\beta_\infty = y$. Thus, β_∞ has to simultaneously satisfy

$$X\beta_\infty = y \quad \text{and} \quad \beta_\infty = b_{\alpha_{\text{n-SAM}}}(X^\top \nu),$$

where $b_\alpha(z) = 2\alpha^2 \odot \sinh(z)$ and $\nu = -\frac{2}{n}\int_0^\infty r_{sam}(s)ds$, and

$$\alpha_{\text{n-SAM}} = \alpha \odot \exp\left(-\frac{2\rho}{n^2}\int_0^\infty (X^\top r_{sam}(s)) \odot (X^\top r(s))ds\right). \tag{5.14}$$

Next we combine the flow expression $b^{-1}_{\alpha_{\text{n-SAM}}}(\beta_\infty) = X^\top \nu$ with a KKT condition $\nabla\phi_\alpha(w) = X^\top\nu$ and get that

$$\nabla\phi_\alpha(\beta) = b_\alpha^{-1}(\beta) = \operatorname{arcsinh}\left(\frac{1}{2\alpha^2} \odot \beta\right).$$

Integration of this equation leads to $\phi_\alpha(\beta) = \sum_{i=1}^d \alpha_i^2 q(\beta_i/\alpha_i^2)$ where $q(z) = \int_0^z \operatorname{arcsinh}(u/2)du = 2 - \sqrt{4+z^2} + z\operatorname{arcsinh}(z/2)$, i.e., exactly the potential function defined in Eq. (5.12). Thus, we conclude that β_∞ satisfies the KKT conditions $X\beta_\infty = y$ and $\nabla\phi_\alpha(\beta_\infty) = X^\top\nu$ for the minimum norm interpolator problem:

$$\min_{\beta\in\mathbb{R}^d} \phi_\alpha(\beta) \quad \text{s.t.} \quad X\beta = y,$$

which proves the first part of the result.

Now to get the expression for $\alpha_{\text{n-SAM}}$, we apply the definition of $r_{sam}(s)$ and obtain

$$\begin{aligned}
r_{sam}(t) &= \tilde{X} w_{sam}(t)^2 - y \\
&= \tilde{X}\left(w(t) + \frac{\rho}{n}\left(\tilde{X}^\top r(t)\right) \odot w(t)\right)^2 - y \\
&= r(t) + \frac{2\rho}{n}\tilde{X}\left(\tilde{X}^\top r(t)\right) \odot w(t) + \frac{\rho^2}{n^2}\tilde{X}\left(\tilde{X}^\top r(t)\right)^2 \odot w(t)^2 \\
&= r(t) + \frac{2\rho}{n}X\left(X^\top r(t)\right) \odot (w_+(t) + w_-(t)) + \frac{\rho^2}{n^2}X\left(X^\top r(t)\right)^2 \odot (w_+(t)^2 + w_-(t)^2).
\end{aligned}$$

Thus we conclude that $X^\top r_{sam}(t) = X^\top r(t) + O(\rho)$ which we plug in Eq. (5.14) to obtain the second part of the theorem:

$$\alpha_{\text{n-SAM}} = \alpha \odot \exp\left(-\frac{2\rho}{n^2}\int_0^\infty (X^\top r_s)^2 ds + O(\rho^2)\right).$$

□

5.9.2 Implicit Bias of the 1-SAM Algorithm

We characterize similarly the implicit bias of the 1-SAM dynamics (5.11) in the following theorem using the function ϕ_α defined in Eq. (5.12).

Theorem 5.9.3. *If the solution β_∞ of the 1-SAM gradient flow (5.11) started from $w_+ = w_- = \alpha \in \mathbb{R}^d_{>0}$ for the squared parameter problem in Eq. (5.5) satisfies $X\beta_\infty = y$, then*

$$\beta_\infty = \arg\min_\beta \phi_{\alpha_{1\text{-SAM}}}(\beta) \quad s.t. \quad X\beta = y,$$

where $\alpha_{1\text{-}SAM} = \alpha \odot \exp\left(-\frac{8\rho}{n}\int_0^\infty \sum_{i=1}^n x_i^2(x_i^\top \beta(s) - y_i)^2 ds + O(\rho^2)\right)$.

In addition, assume that there exist $R, B \geq 0$ *such that almost surely (1) the inputs are bounded* $\|x\|_2 \leq R$ *and (2) the trajectory of the flow is bounded* $\|\beta(t)\|_2 \leq B$ *for all* $t \geq 0$. *Then for all* $\rho \leq \frac{1}{4R^2\sqrt{B(B+\|\beta_*\|_2)}}$, *we have that* $\alpha_{1\text{-}SAM,i} \leq \alpha_i$ *for* $i \in \{1, \dots, d\}$.

Proof. The proof follows the same lines as the proof of Theorem 5.9.2. We denote a concatenation of positive and negative copies of the i-th training example as $\tilde{x}_i = \left[\begin{smallmatrix} x_i \\ -x_i \end{smallmatrix}\right] \in \mathbb{R}^{2d}$, the intermediate step of 1-SAM based on the i-th training example as $w_{sam}^{(i)}(t) \in \mathbb{R}^d$, the residuals of $w(t)$ and $w_{sam}^{(i)}(t)$ on the i-th training example as $r_i(t) = \tilde{x}_i^\top w(t)^2 - y_i$ and $r_{sam,i}(t) = \tilde{x}_i^\top w_{sam}^{(i)}(t)^2 - y_i$. Then we have that the dynamics of the flow (5.11) satisfies

$$\begin{aligned}
\dot{w}(t) &= -\frac{1}{n}\sum_{i=1}^n \nabla \ell_i(w_{sam}^{(i)}(t)) \\
&= -\frac{1}{n}\sum_{i=1}^n r_{sam,i}(t) \cdot \tilde{x}_i \odot w_{sam}^{(i)}(t) \\
&= -\frac{1}{n}\sum_{i=1}^n r_{sam,i}(t) \cdot \tilde{x}_i \odot w(t) \odot (\mathbf{1} + 4\rho r_i(t)\tilde{x}_i)\,.
\end{aligned}$$

Integration of this ODE leads to

$$w(t) = w(0) \odot \exp\left(-\frac{1}{n}\tilde{X}^\top \int_0^t r_{sam}(s)ds\right) \odot \exp\left(-\frac{4\rho}{n}\sum_{i=1}^n \tilde{x}_i^2 \int_0^t r_{sam,i}(s) r_i(s) ds\right).$$

The rest of the proof is similar to the one of Theorem 5.9.2 and we directly obtain that

$$\alpha_{1\text{-SAM}} = \alpha \odot \exp\left(-\frac{8\rho}{n}\sum_{i=1}^n \tilde{x}_i^2 \int_0^t r_{sam,i}(s) r_i(s) ds\right). \tag{5.15}$$

Using the definition of $r_{sam,i}(t)$ we have

$$\begin{aligned}
r_{sam,i}(t) &= \tilde{x}_i^\top w_{sam}(t)^2 - y_i \\
&= \tilde{x}_i^\top w(t)^2 \odot (\mathbf{1} + 4\rho r_i(t)\tilde{x}_i)^2 - y_i \\
&= \tilde{x}_i^\top w(t)^2 \odot \left(\mathbf{1} + 8\rho r_i(t)\tilde{x}_i + 16\rho^2 r_i(t)^2 \tilde{x}_i^2\right) - y_i \\
&= r_i(t) + 8\rho r_i(t)\left(w_+(t)^2 + w_-(t)^2\right)^\top x_i^2 + 16\rho^2 r_i(t)^2 \left(w_+(t)^2 - w_-(t)^2\right)^\top x_i^3 \\
&= r_i(t) + 8\rho r_i(t)\left(w_+(t)^2 + w_-(t)^2\right)^\top x_i^2 + 16\rho^2 r_i(t)^2 \beta(t)^\top x_i^3
\end{aligned}$$

And therefore

$$x_i^2 r_{sam,i}(t) r_i(t) = r_i(t)^2 x_i^2 \odot \left(\mathbf{1} + 8\rho\left(w_+(t)^2 + w_-(t)^2\right)^\top x_i^2 + 16\rho^2 r_i(t)\beta(t)^\top x_i^3\right) \tag{5.16}$$

This leads to the result stated in the theorem

$$\alpha_{\text{1-SAM}} = \alpha \odot \exp\left(-\frac{8\rho}{n}\int_0^\infty \sum_{i=1}^n x_i^2(x_i^\top \beta(s) - y_i)^2 ds + O(\rho^2)\right). \tag{5.17}$$

Additionally, from Eq. (5.16) we can conclude that having ρ such that $1+16\rho^2 r_i(t)\beta(t)^\top x_i^3 \geq 0$ is sufficient to guarantee that $\alpha_{\text{1-SAM},i} \leq \alpha_i$ for every i. We can use Cauchy-Schwarz inequality twice to upper bound $|r_i(t)\beta(t)^\top x_i^3|$:

$$\begin{aligned}|r_i(t)\beta(t)^\top x_i^3| = |x_i^\top(\beta-\beta_*)\beta(t)^\top x_i^3| &\leq \|x_i\|_2\|\beta(t)-\beta_*\|_2\|\beta(t)\|_2\|x_i^3\|_2 \\ &\leq \|x_i\|_2^4(\|\beta(t)\|_2+\|\beta_*\|_2)\|\beta(t)\|_2 \leq R^4(B+\|\beta_*\|_2)B\end{aligned}$$

Thus, we have that $\rho^2 r_i(t)\beta(t)^\top x_i^3 \geq -\rho^2 R^4(B+\|\beta_*\|_2)B \geq -\frac{1}{16}$ which leads to the upper bound stated in the theorem $\rho \leq \frac{1}{4R^2\sqrt{B(B+\|\beta_*\|_2)}}$. □

5.9.3 Comparison between 1-SAM and n-SAM

Theoretical comparison. We wish to compare the two leading terms of the exponents in $\alpha_{\text{n-SAM}}$ and $\alpha_{\text{1-SAM}}$:

$$I_{\text{n-SAM}}(t) = \frac{1}{n^2}\left(X^\top r(t)\right)^2 = \frac{1}{n^2}\left(\sum_{i=1}^n x_i r_i(t)\right)^2 \quad \text{and} \quad I_{\text{1-SAM}}(t) = \frac{1}{n}\sum_{i=1}^n x_i^2 r_i(t)^2,$$

and relate them to the loss values at $w(t)$.

We first note that using Cauchy-Schwarz inequality can directly imply that $I_{\text{1-SAM},i}(t) \geq I_{\text{n-SAM},i}(t)$. However, we aim at obtaining a more quantitative result, even though the following derivations will be informal. Comparing the ℓ_1-norms of $I_{\text{n-SAM}}(t)$ and $I_{\text{1-SAM}}(t)$ amounts to compare the following two quantities:

$$\begin{aligned}\|I_{\text{n-SAM}}(t)\|_1 &= (w(t)-w_*)^\top\left[\frac{1}{n}\sum_{i=1}^n x_i x_i^\top\right]^2(w(t)-w_*), \\ \|I_{\text{1-SAM}}(t)\|_1 &= (w(t)-w_*)^\top\left[\frac{1}{n}\sum_{i=1}^n \|x_i\|_2^2 x_i x_i^\top\right](w(t)-w_*).\end{aligned}$$

We can compare the typical operator norms of the random matrices that define the two quadratic forms. If we assume that $x_i \sim \mathcal{N}(0, I_d)$, then following the Bai-Yin's law, the operator norm of a Wishart matrix is with high probability $\|\frac{1}{n}\sum_{i=1}^n x_i x_i^\top\|_{op} \approx \frac{d}{n}$ and that with high probability, the squared norm of a Gaussian vector is $\|x_i\|_2^2 \approx d$. Therefore we obtain that

$$\left\|\left[\frac{1}{n}\sum_{i=1}^n x_i x_i^\top\right]^2\right\|_{op} = \left\|\frac{1}{n}\sum_{i=1}^n x_i x_i^\top\right\|_{op}^2 \approx \frac{d^2}{n^2},$$

$$\left\| \frac{1}{n} \sum_{i=1}^{n} \|x_i\|^2 x_i x_i^\top \right\|_{op} \approx d \left\| \frac{1}{n} \sum_{i=1}^{n} x_i x_i^\top \right\|_{op} \approx \frac{d^2}{n}.$$

Therefore in the overparametrized regime ($d >> n$), we typically have that $\frac{\|I_{\text{1-SAM}}(t)\|_1}{\|I_{\text{n-SAM}}(t)\|_1} \approx n$ and the biasing effect of 1-SAM would tend to be $O(n)$ times better compared to n-SAM.

However, this first insight only enables to compare $I_{\text{n-SAM}}(t)$ and $I_{\text{1-SAM}}(t)$. It is not informative on the intrinsic biasing effect of n-SAM and 1-SAM. With this aim, we would like to relate the quantities $I_{\text{n-SAM}}(t)$ and $I_{\text{1-SAM}}(t)$ to the loss function evaluated in $w(t)$. Using the concentration of Wishart matrices, i.e., $\frac{1}{d}[XX^\top] \approx I$ for large dimension d, we have with high probability

$$\begin{aligned}
\|I_{\text{n-SAM}}(t)\|_1 &= \frac{1}{n^2}(w(t) - w_*)^\top X^\top X X^\top X (w(t) - w_*) \\
&= \frac{d}{n^2}(w(t) - w_*)^\top X^\top \frac{1}{d}[XX^\top] X (w(t) - w_*) \\
&\approx \frac{d}{n}(w(t) - w_*)^\top \frac{1}{n}[X^\top X](w(t) - w_*) \\
&= \frac{d}{n} L(w(t)). \qquad (5.18)
\end{aligned}$$

And using the concentration of Gaussian vectors, we also have that

$$\begin{aligned}
\|I_{\text{1-SAM}}(t)\|_1 &= (w(t) - w_*)^\top \frac{1}{n} \sum_{i=1}^{n} \|x_i\|^2 x_i x_i^\top (w(t) - w_*) \\
&\approx d(w(t) - w_*)^\top \frac{1}{n} \sum_{i=1}^{n} x_i x_i^\top (w(t) - w_*) \\
&= dL(w(t)). \qquad (5.19)
\end{aligned}$$

These approximations provide some intuition on why the biasing effect of 1-SAM and n-SAM can be related to the integral of the loss and that typically the difference is on the order of n. We let a formal derivation of these results as future work.

Experiments with stochastic ERM, n-SAM, 1-SAM. We provide an additional experiment to investigate the performance of stochastic implementations of the ERM, n-SAM and 1-SAM. As explained by Pesme et al. (2021), we observe in Fig. 5.13 that the stochastic implementations enjoy a better implicit bias than their deterministic counterparts. We note that the fact that small batch versions generalize better than full batch version is commonly observed in practice for deep networks Keskar et al. (2016). We let the characterization of the implicit bias of these stochastic implementations as future works.

Grid search over ρ for n-SAM vs. 1-SAM. We note that for Fig. 5.6 and Fig. 5.13, we used a fixed ρ which was the same for both n-SAM and 1-SAM. Tuning ρ for each method separately can help to achieve a better test loss for both methods as shown in Fig. 5.14. We can see that 1-SAM still significantly outperforms ERM and n-SAM for the

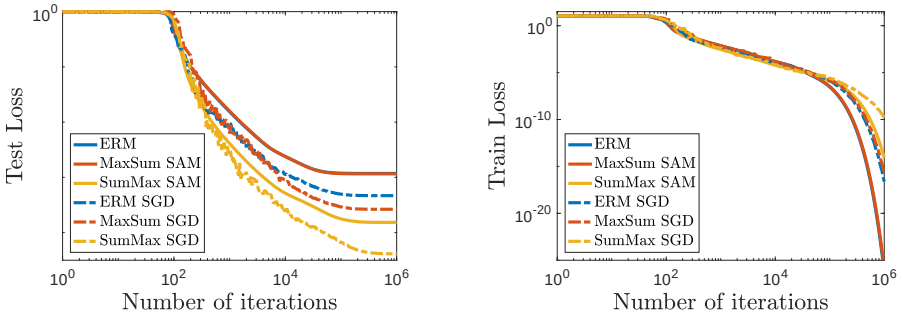


Figure 5.13: Implicit bias of SAM on a sparse regression problem using a diagonal linear network with $d = 30$, $n = 20$, $x_i \sim \mathcal{N}(0, I)$, $\kappa = \|\beta_*\|_0 = 3$, $y_i = x_i^\top \beta_*$. All methods are initialized at $\alpha = 0.01$ and used with step size $\gamma = 1/d$ and $\rho = 1/d$. We can see that 1-SAM (SumMax) SGD converges to a solution which generalizes better (left plot) and enjoys a different implicit bias from the other methods. At the same time, all algorithms converge to a global minimum of f at linear rate (right plot). The convergence speed is inversely proportional to the biasing effect.

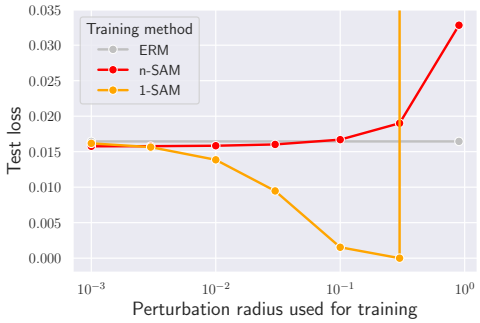
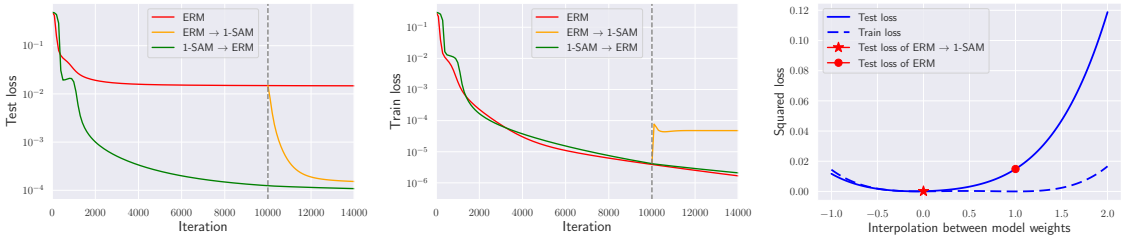


Figure 5.14: A grid search over ρ for full-batch n-SAM vs. 1-SAM ($\alpha = 0.05$, $\gamma = 15/d$ for all methods). We can see that even with the optimal ρ, n-SAM generalizes much worse than 1-SAM which is coherent with our deep learning experiments in Fig. 5.1.

optimally chosen radius ρ and that n-SAM leads only to marginal improvements.

Connection to the ERM $\to$ SAM and SAM $\to$ ERM experiment. Here we provide further details on the connection between Theorem 5.5.1 and the empirical results in Fig. 5.9. First of all, we show in Fig. 5.15 that the same observations as we observed for deep networks also hold on a diagonal linear network. In this experiment, we used the initialization scale $\alpha = 0.05$, $\rho_{\text{1-SAM}} = 0.175$, and $\rho_{\text{GD}\to\text{1-SAM}} = 10.0$. We note that we had to take $\rho_{\text{GD}\to\text{1-SAM}}$ significantly larger than $\rho_{\text{1-SAM}}$ since after running GD, we are already near a global minimum where the gradients (which are also used for the ascent step of SAM) are very small so we need to increase the inner step size $\rho_{\text{GD}\to\text{1-SAM}}$ to observe a difference. In addition, a loss interpolation between $w_{\text{GD}\to\text{1-SAM}}$ and w_{GD} reveals linear connectivity between the two found minima suggesting that both minima are situated in the same asymmetric basin, similarly to what we observed for deep networks in Fig. 5.10.

First we note that Theorem 5.5.1 can be trivially adapted to the case where SAM is used with varying inner step size ρ_t, and would therefore show that for diagonal linear networks, the key quantity determining the magnitude of the implicit bias for SAM



(a) (a) Test loss over epochs **(b)** (b) Training loss over epochs **(c)** (c) Loss interpolations

Figure 5.15: Test loss (a) and training loss (b) for full-batch ERM compared to ERM → 1-SAM and 1-SAM → ERM on a diagonal linear network where we switch between the methods after 10k iterations. We can see that 1-SAM can quickly escape the worse-generalizing minimum found by ERM. Moreover, in (c) we show loss interpolations between ERM → 1-SAM and ERM that show that they are linearly connected and situated in the same basin.

is the integral of the step size ρ_s times the loss over the optimization trajectory $w(s)$, i.e., $\|\Delta_{\text{1-SAM-}\rho_s}\|_1 \approx d\int_0^\infty \rho_s L(w(s))ds$ which leads to a smaller value in the exponent $\alpha_{\text{1-SAM-}\rho_s} = \alpha e^{-\rho\Delta_{\text{1-SAM-}\rho_s}+O(\rho^2)}$, thus decreasing the effective α and biasing the flow to a sparser solution.

In the case of ERM → 1-SAM, it amounts to consider a step size $\rho_s = 0$ if $s < t$ and $\rho_s = \rho$ after the switch. Therefore the integral is taken only over the last epochs, and $\|\Delta_{\text{1-SAM-t-}\infty}\|_1 \approx d\int_t^\infty L(w(s))ds$ where the integral starts at the time step t. The resulting $\|\Delta_{\text{1-SAM-t-}\infty}\|_1$ is smaller than $\|\Delta_{\text{1-SAM}}\|_1$ but it can still be sufficient (especially, when using a higher ρ as we do for Fig. 5.15) to improve the biasing effect so that it leads to noticeable improvements in generalization.

At the same time, for 1-SAM → ERM, which amounts to consider a step size $\rho_s = \rho$ if $s < t$ and $\rho_s = 0$ after the switch, the integral is already large enough due to the first 1000 epochs with SAM, leading to a term $\|\Delta_{\text{1-SAM-0-t}}\|_1 \approx d\int_0^t L(w(s))ds$ and switching back to ERM preserves the implicit bias due to a low enough effective α. This explains why switching back to ERM does not negatively affect generalization of the model.

5.10 Convergence of the SAM Algorithm

In this section we provide proofs of convergence for SAM. We consider first the full-batch SAM algorithm and then its stochastic version.

5.10.1 Convergence of Full-Batch n-SAM

We first consider the full-batch version of SAM, i.e., the following update rule:

$$w_{t+1} = w_t - \gamma\nabla L\left(w_t + \rho\nabla L(w_t)\right). \tag{5.20}$$

We note that this update rule is reminiscent of the extra-gradient algorithm (Korpelevich, 1977) but with an *ascent* in the inner step instead of a *descent*. Moreover, this update rule can also be seen as a realization of the general extrapolated gradient descent framework suggested in Lin et al. (2020). However, taking an *ascent* step for extrapolation is not discussed there, and the convergence properties of the update rule from Eq. (5.20), to the best of our knowledge, have not been proven.

Summary of the convergence results. Let us first recall the definition of β-smoothness which we will use in our proofs.

(A2') (β-smoothness). *There exists* $\beta > 0$ *such that* $\|\nabla L(w) - \nabla L(v)\| \leq \beta \|w - v\|$ *for all* $w, v \in \mathbb{R}^d$.

When the function L is β-smooth, convergence to stationary points can be obtained.

Theorem 5.10.1. *Assume* ***(A2')****. For any* $\gamma < 1/\beta$ *and* $\rho < 1/\beta$*, the iterates (5.20) satisfy for all* $T \geq 0$*:*

$$\frac{1}{T} \sum_{t=0}^{T-1} \|\nabla L(w_t)\|^2 \leq \frac{2}{\gamma(1 - \rho\beta)T} (L(w_0) - L_*),$$

If, in addition, the function L *satisfies* ***(A3)****, then:*

$$L(w_T) - L_* \leq \left(1 - \frac{\gamma(1 - \rho\beta)\mu}{2}\right)^T (L(w_0) - L_*).$$

We can make the following remarks:

- We recover the rates of gradient descent but with constants increasing with the ascent step size ρ.
- The condition $\rho < 1/\beta$ is necessary since the point $w + 1/\beta \nabla L(w)$ can be a local maximum of L. Such w would be a fixed point of the algorithm without being a stationary point of L.
- The proof crucially relies on the bound $\langle \nabla L(w_t + \rho \nabla L(w_t)), \nabla L(w_t) \rangle \geq (1 - \rho\beta) \|\nabla L(w_t)\|^2$ which shows that the SAM step is well-aligned with the gradient step (see Lemma 3) and on a descent inequality similar to the classical one for gradient descent (see Lemma 4).
- For non-convex functions, full details are provided in Theorem 5.10.2. When the function satisfies in addition Polyak-Lojasiewicz inequality, a stronger result holds which is stated in Theorem 5.10.3.
- For convex functions, $\langle \nabla L(w_t + \rho \nabla L(w_t)), \nabla L(w_t) \rangle \geq \|\nabla L(w_t)\|^2$ and convergence holds for any step size ρ given that $\gamma\rho$ is small enough. Details are provided in Theorem 5.10.4.

Auxiliary Lemmas. The following lemma shows that the SAM update is well correlated with the gradient $\nabla L(w)$ and will be a cornerstone to our proof.

Lemma 3. *Let L be a differentiable function and $w \in \mathbb{R}^d$. We have the following bound for any $\rho \geq 0$:*

$$\langle \nabla L(w + \rho \nabla L(w)), \nabla L(w) \rangle \geq (1 + \alpha\rho) \|\nabla L(w)\|^2 \text{ where } \alpha = \begin{cases} -\beta & \text{if } L \text{ is } \beta\text{-smooth,} \\ 0 & \text{if } L \text{ is convex} \\ \mu & \text{if } L \text{ is } \mu\text{-strongly convex.} \end{cases}$$

Proof. We simply add and subtract a term $\|\nabla L(w)\|^2$ in order to make use of classical inequalities bounding $\langle \nabla L(w_1) - \nabla L(w_2), w_1 - w_2 \rangle$ by $\|w_1 - w_2\|^2$ for smooth or convex functions and $w_1, w_2 \in \mathbb{R}^d$.

$$\begin{aligned} \langle \nabla L(w + \rho \nabla L(w)), \nabla L(w) \rangle &= \langle \nabla L(w + \rho \nabla L(w)) - \nabla L(w), \nabla L(w) \rangle) + \|\nabla L(w)\|^2 \\ &= 1/\rho \langle \nabla L(w + \rho \nabla L(w)) - \nabla L(w), \rho \nabla L(w) \rangle + \|\nabla L(w)\|^2 \\ &\geq (1 + \alpha\rho) \|\nabla L(w)\|^2, \end{aligned}$$

where the last inequality is using that

$$\langle \nabla L(w_1) - \nabla L(w_2), w_1 - w_2 \rangle \geq \alpha \|w_2 - w_1\|^2, \text{ where } \alpha = \begin{cases} -\beta & \text{if } L \text{ is } \beta\text{-smooth,} \\ 0 & \text{if } L \text{ is convex} \\ \mu & \text{if } L \text{ is } \mu\text{-strongly convex.} \end{cases}$$

□

The next lemma shows that the decrease of function values of the SAM algorithm defined in Eq. (5.20) can be controlled similarly as in the case of gradient descent (Nesterov, 2004).

Lemma 4. *Assume **(A2')**. For any $\gamma \leq 1/\beta$, the iterates (5.20) satisfy for all $t \geq 0$:*

$$L(w_{t+1}) \leq L(w_t) - \gamma(1 - \rho\beta)\Big(1 - \frac{\gamma\beta}{2}(1 - \rho\beta)\Big)\|\nabla L(w_t)\|^2.$$

*If, in addition, the function L satisfies **(A3)** with potentially $\mu = 0$, then for all $\gamma, \rho \geq 0$ such that $\gamma\beta(2 - \rho\beta) \leq 2$, we have*

$$L(w_{t+1}) \leq L(w_t) - \gamma\Big(1 - \frac{\gamma\beta}{2} + \rho\mu\Big(1 - \gamma\beta - \frac{\gamma\rho\beta^2}{2}\Big)\Big)\|\nabla L(w_t)\|^2.$$

We note that the constraints on the step size are different depending on the assumptions on the function L. In the non-convex case, ρ has to be smaller than $1/\beta$, whereas in the convex case, it has to be smaller than $2/\beta$.

Proof. Let us define by $w_{t+1/2} = w_t + \rho \nabla L(w_t)$ the SAM ascent step. Using the smoothness of the function L (Assumption **(A2')**), we obtain

$$L(w_{t+1}) \leq L(w_t) - \gamma \langle \nabla L(w_{t+1/2}), \nabla L(w_t) \rangle + \frac{\gamma^2 \beta}{2} \|\nabla L(w_{t+1/2})\|^2.$$

The main trick is to use the binomial squares

$$\|\nabla L(w_{t+1/2})\|^2 = -\|\nabla L(w_t)\|^2 + \|\nabla L(w_{t+1/2}) - \nabla L(w_t)\|^2 + 2\langle \nabla L(w_{t+1/2}), \nabla L(w_t) \rangle,$$

to bound

$$\begin{aligned}
L(w_{t+1}) &\leq L(w_t) - \gamma \langle \nabla L(w_{t+1/2}), \nabla L(w_t) \rangle + \frac{\gamma^2 \beta}{2} \|\nabla L(w_{t+1/2})\|^2 \\
&= L(w_t) - \frac{\gamma^2 \beta}{2} \|\nabla L(w_t)\|^2 + \frac{\gamma^2 \beta}{2} \|\nabla L(w_{t+1/2}) - \nabla L(w_t)\|^2 - \gamma(1 - \gamma\beta) \langle \nabla L(w_{t+1/2}), \nabla L(w_t) \rangle \\
&\leq L(w_t) - \gamma[1 - \rho\beta - \frac{\gamma\beta}{2}(1 - \rho\beta)^2] \|\nabla L(w_t)\|^2,
\end{aligned}$$

where we have used Lemma 3 and that $\|\nabla L(w_{t+1/2}) - \nabla L(w_t)\|^2 \leq \beta^2 \|w_{t+1/2} - w_t\|^2 \leq \beta^2 \rho^2 \|\nabla L(w_t)\|^2$.

If, in addition, the function L is convex then we can use its co-coercivity (Nesterov, 2004) to bound $\|\nabla L(w_{t+1/2}) - \nabla L(w_t)\|^2 \leq \beta \langle \nabla L(w_{t+1/2}) - \nabla L(w_t), w_{t+1/2} - w_t \rangle$ and obtain a tighter bound:

$$\begin{aligned}
L(w_{t+1}) &\leq L(w_t) - \gamma \langle \nabla L(w_{t+1/2}), \nabla L(w_t) \rangle + \frac{\gamma^2 \beta}{2} \|\nabla L(w_{t+1/2})\|^2 \\
&= L(w_t) - \frac{\gamma^2 \beta}{2} \|\nabla L(w_t)\|^2 + \frac{\gamma^2 \beta}{2} \|\nabla L(w_{t+1/2}) - \nabla L(w_t)\|^2 - \gamma(1 - \gamma\beta) \langle \nabla L(w_{t+1/2}), \nabla L(w_t) \rangle \\
&\leq L(w_t) - \gamma(1 - \frac{\gamma\beta}{2}) \|\nabla L(w_t)\|^2 - \gamma(1 - \gamma\beta - \frac{\gamma\rho\beta^2}{2}) \langle \nabla L(w_{t+1/2}) - \nabla L(w_t), \nabla L(w_t) \rangle \\
&\leq L(w_t) - \gamma(1 - \frac{\gamma\beta}{2} + \rho\mu(1 - \gamma\beta - \frac{\gamma\rho\beta^2}{2})) \|\nabla L(w_t)\|^2,
\end{aligned}$$

where we have used Lemma 3. □

Convergence proofs. Using the previous Lemma 4 recursively, we can bound the average gradient value of the iterates (5.20) of SAM algorithm and ensure convergence to stationary points.

Theorem 5.10.2. *Assume **(A2')**. For any $\gamma < 1/\beta$ and $\rho < 1/\beta$, the iterates (5.20) satisfies for all $T \geq 0$:*

$$\frac{1}{T} \sum_{t=0}^{T} \|\nabla L(w_t)\|^2 \leq \frac{L(w_0) - L(w_T)}{T\gamma(1 - \rho\beta)[1 - \frac{\gamma\beta}{2}(1 - \rho\beta)]}.$$

Proof. Using the Lemma 4 we obtain

$$\gamma(1-\rho\beta)\Big(1-\frac{\gamma\beta}{2}(1-\rho\beta)\Big)\|\nabla L(w_t)\|^2 \leq L(w_t) - L(w_{t+1}).$$

And summing these inequalities for $t = 0, \ldots, T-1$ yields

$$\frac{1}{T}\sum_{t=0}^{T-1}\|\nabla L(w_t)\|^2 \leq \frac{L(w_0) - L(w_T)}{T\gamma(1-\rho\beta)[1-\frac{\gamma\beta}{2}(1-\rho\beta)]}.$$

□

When the function L additionally satisfies a Polyak-Lojasiewicz condition **(A3)**, linear convergence of the function value to the minimum function value can be obtained. This is the object of the following theorem:

Theorem 5.10.3. *Assume **(A2')** and **(A3)**. For any $\gamma < 1/\beta$ and $\rho < 1/\beta$, the iterates (5.20) satisfies for all $T \geq 0$:*

$$L(w_t) - L_* \leq \left(1 - 2\gamma\mu(1-\rho\beta)\Big(1-\frac{\gamma\beta}{2}(1-\rho\beta)\Big)\right)^t (L(w_0) - L_*).$$

Proof. Using the Lemma 4 and that the function L is μ Polyak-Lojasiewicz (Assumption **(A3)**) we obtain

$$L(w_{t+1}) \leq L(w_t) - 2\mu\gamma(1-\rho L)\Big(1-\frac{\gamma\beta}{2}(1-\rho L)\Big)(L(w_t) - L_*).$$

And subtracting the optimal value L_* we get

$$\begin{aligned} L(w_t) - L_* &\leq \left(1 - 2\gamma\mu(1-\rho\beta)\Big(1-\frac{\gamma\beta}{2}(1-\rho\beta)\Big)\right)(L(w_{t-1}) - L_*) \\ &\leq \left(1 - 2\gamma\mu(1-\rho\beta)\Big(1-\frac{\gamma\beta}{2}(1-\rho\beta)\Big)\right)^t (L(w_0) - L_*). \end{aligned}$$

□

When the function L is convex, convergence of the average of the iterates can be proved.

Theorem 5.10.4. *Assume **(A2')** and L convex. For any step sizes γ and ρ such that $\gamma\beta(1+\rho\beta) < 2$, then the averaged $\bar{w}_T = \frac{1}{T}\sum_{t=0}^{T-1} w_t$ of the iterates (5.20) satisfies for all $T \geq 0$:*

$$L(\bar{w}_T) - L_* \leq \frac{2\rho\beta + 1}{\gamma(2 - \gamma\beta(1+\rho\beta))T}\|w_0 - w_*\|^2,$$

If, in addition, the function L is μ-strongly convex, then:

$$\|w_T - w_*\|^2 \leq \Big(1 - \gamma\mu(2 - \gamma\beta(1+\rho\beta))\Big)^T (2\rho + 1)\|w_0 - w_*\|^2.$$

The proof is using a different astute Lyapunov function which works for the non-strongly convex case.

Proof. Let us define by $V_t = [L(w_t) - L(w_*)] + \frac{1}{2\rho}\|w_t - w_*\|^2$ and by $w_{t+1/2} = w_t + \rho\nabla L(w_t)$ the SAM ascent step.

$$\begin{aligned}
V_{t+1} - V_t &\leq -\frac{\gamma}{\rho}\langle \nabla L(w_{t+1/2}), w_t - w_*\rangle - \gamma\langle \nabla L(w_{t+1/2}), \nabla L(w_t)\rangle + \frac{\gamma^2}{2\rho}(1+\rho\beta)\|\nabla L(w_{t+1/2})\|^2 \\
&= -\frac{\gamma}{\rho}\langle \nabla L(w_{t+1/2}), w_t + \rho\nabla L(w_t) - w_*\rangle + \frac{\gamma^2}{2\rho}(1+\rho\beta)\|\nabla L(w_{t+1/2})\|^2 \\
&= -\frac{\gamma}{\rho}\langle \nabla L(w_{t+1/2}), w_{t+1/2} - w_*\rangle + \frac{\gamma^2}{2\rho}(1+\rho\beta)\|\nabla L(w_{t+1/2})\|^2 \\
&\leq -\frac{\gamma}{\rho}(1 - \frac{\gamma\beta}{2}(1+\rho\beta))\langle \nabla L(w_{t+1/2}), w_{t+1/2} - w_*\rangle.
\end{aligned}$$

If L is convex then $L(w_{t+1/2}) - L(w_*) \leq \langle \nabla L(w_{t+1/2}), w_{t+1/2} - w_*\rangle$ and therefore we obtain

$$\frac{\gamma}{\rho}\left(1 - \frac{\gamma\beta}{2}(1+\rho\beta)\right)\Big(L(w_{t+1/2}) - L(w_*)\Big) \leq V_t - V_{t+1}.$$

Using the definition of $w_{t+1/2}$ we always have that $L(w_{t+1/2}) \geq L(w_t) + \rho\|\nabla L(w_t)\|^2$ therefore

$$\frac{\gamma}{\rho}\left(1 - \frac{\gamma\beta}{2}(1+\rho\beta)\right)(L(w_t) - L(w_*)) \leq V_t - V_{t+1}.$$

And taking the sum and using Jensen inequality we finally obtain:

$$L(\frac{1}{T}\sum_{t=0}^{T} w_t) - L(w_*) \leq \frac{V_0 - V_{T+1}}{T\frac{\gamma}{\rho}(1 - \frac{\gamma\beta}{2}(1+\rho\beta))}.$$

If L is μ-strongly convex, we use that $\langle \nabla L(w_{t+1/2}), w_{t+1/2} - w_*\rangle \geq \mu\|w_{t+1/2} - w_*\|^2$ to obtain

$$\begin{aligned}
\|w_{t+1/2} - w_*\|^2 = \|w_t + \rho\nabla L(w_t) - w_*\|^2 &= \|w_t - w_*\|^2 + 2\rho\langle \nabla L(w_t), w_t - w_*\rangle + \rho^2\|\nabla L(w_t)\|^2 \\
&\geq \|w_t - w_*\|^2 + 2\rho\langle \nabla L(w_t), w_t - w_*\rangle \\
&\geq \|w_t - w_*\|^2 + 2\rho[L(w_t) - L(w_*)] \\
&\geq 2\rho V_t.
\end{aligned}$$

Therefore we have

$$V_{t+1} \leq (1 - \gamma\mu(2 - \gamma\beta(1+\rho\beta)))\, V_t \leq (1 - \gamma\mu(2 - \gamma\beta(1+\rho\beta)))^{t+1}\, V_0.$$

□

5.10.2 Convergence of Stochastic SAM

Convergence of n-SAM

When the SAM algorithm is implemented with the n-SAM objective as optimization objective, two different batches are used in the ascent and descent steps. We obtain the n-SAM algorithm defined as

$$w_{t+1} = w_t - \frac{\gamma_t}{b} \sum_{i \in I_t} \nabla \ell_i \Big(w_t + \frac{\rho_t}{b} \sum_{i \in J_t} \nabla \ell_i(w_t) \Big), \tag{5.21}$$

where I_t and J_t are two *different* mini-batches of data of size b. For this variant of the SAM algorithm, we obtain the following convergence result.

Theorem 5.10.5. *Assume **(A1)**, **(A2')** for the iterates (5.21). For any $T \geq 0$ and for step sizes $\gamma_t = \frac{1}{\sqrt{T}\beta}$ and $\rho_t = \frac{1}{T^{1/4}\beta}$, we have:*

$$\frac{1}{T} \mathbb{E} \left[\sum_{t=0}^{T-1} \|\nabla L(w_t)\|^2 \right] \leq \frac{4}{\beta\sqrt{T}} (L(w_0) - L_*) + \frac{8\sigma^2}{b\sqrt{T}},$$

*In addition, under **(A2)**, with step sizes $\gamma_t = \min\{\frac{8t+4}{3\mu(t+1)^2}, \frac{1}{2\beta}\}$ and $\rho_t = \sqrt{\gamma_t/\beta}$:*

$$\mathbb{E}\left[L(w_T)\right] - L_* \leq \frac{3\beta^2 (L(w_0) - L_*)}{\mu^2 T^2} + \frac{22\beta\sigma^2}{b\mu^2 T}$$

We obtain the same convergence result as in Theorem 5.6.1, but under the relaxed smoothness assumption **(A2')**.

As in the deterministic case, the proof relies on two lemmas which shows that the SAM update is well correlated with the gradient and that the decrease of function values can be controlled.

Auxiliary lemmas. The following lemma shows that the SAM update is well correlated with the gradient $\nabla L(w_t)$. Let us denote by $\nabla L_{t+1}(w) = \frac{1}{b} \sum_{i \in I_t} \nabla \ell_i(w)$, $\nabla L_{t+1/2}(w) = \frac{1}{b} \sum_{i \in J_t} \nabla \ell_i(w)$, and $w_{t+1/2} = w_t + \rho \nabla L_{t+1/2}(w_t)$ the SAM ascent step.

Lemma 5. *Assume **(A1)** and **(A2)**. Then for all $\rho \geq 0$, $t \geq 0$ and $w \in \mathbb{R}^d$,*

$$\mathbb{E}\langle \nabla L_{t+1}(w + \rho \nabla L_{t+1/2}(w)), \nabla L(w) \rangle \geq (1/2 - \beta\rho) \|\nabla L(w)\|^2 - \frac{\beta^2 \rho^2 \sigma^2}{2}.$$

The proof is similar to the proof of Lemma 3. Only the stochasticity of the noisy gradients has to be taken into account. For this goal, we consider instead the update which would have been obtained without noise, and bound the remainder using the bounded variance assumption **(A1)**.

Proof. Let us denote by $\hat{w} = w + \rho\nabla L(w)$, the true gradient step. We first add and subtract $\nabla L_{t+1/2}(\hat{w})$

$$\langle \nabla L_{t+1}(w + \rho\nabla L_{t+1/2}(w)), \nabla L(w)\rangle = \langle \nabla L_{t+1}(w + \rho\nabla L_{t+1/2}(w)) - \nabla L_{t+1}(\hat{w}), \nabla L(w)\rangle - \langle \nabla L_{t+1}(\hat{w}), \nabla L(w)\rangle.$$

We bound the two terms separately. We use the smoothness of L (Assumption **(A2')**) to bound the first term:

$$\begin{aligned}
&- \mathbb{E}\langle \nabla L_{t+1}(w + \rho\nabla L_{t+1/2}(w)) - \nabla L_{t+1}(\hat{w}), \nabla L(w)\rangle \\
&= - \mathbb{E}\langle \nabla L(w + \rho\nabla L_{t+1/2}(w)) - \nabla L(\hat{w}), \nabla L(w)\rangle \\
&\leq \frac{1}{2}\,\mathbb{E}\,\|\nabla L(w + \rho\nabla L_{t+1/2}(w)) - \nabla L(\hat{w})\|^2 + \frac{1}{2}\|\nabla L(w)\|^2 \\
&\leq \frac{\beta^2}{2}\,\mathbb{E}\,\|w + \rho\nabla L_{t+1/2}(w) - \hat{w}\|^2 + \frac{1}{2}\|\nabla L(w)\|^2 \\
&\leq \frac{\beta^2\rho^2}{2}\,\mathbb{E}\,\|\nabla L_{t+1/2}(w) - \nabla L(w)\|^2 + \frac{1}{2}\|\nabla L(w_t)\|^2 \\
&\leq \frac{\beta^2\rho^2\sigma^2}{2b} + \frac{1}{2}\,\|\nabla L(w)\|^2,
\end{aligned}$$

where we have used that the variance of a mini-batch of size b is bounded by σ^2/b. Note that this term can be equivalently bounded by $\beta\rho\sigma/\sqrt{b}\|\nabla L(w)\|$ if needed. For the second term, we directly apply Lemma 3 to obtain

$$\mathbb{E}\langle \nabla L_{t+1}(\hat{w}), \nabla L(w)\rangle = \mathbb{E}\langle \nabla L(\hat{w}), \nabla L(w)\rangle \geq (1 - \beta\rho)\|\nabla L(w)\|^2.$$

□

The next lemma shows that the decrease of function values of stochastic n-SAM can be controlled similarly as for standard stochastic gradient descent.

Lemma 6. *Let us assume **(A1, A2')** then for all $\gamma \leq \frac{1}{2\beta}$ and $\rho \leq \frac{1}{2\beta}$, the iterates (5.21) satisfies*

$$\mathbb{E}\,L(w_{t+1}) \leq \mathbb{E}\,L(w_t) - \frac{\gamma}{4}\,\mathbb{E}\,\|\nabla L(w_t)\|^2 + \gamma\beta\sigma^2(\gamma + \rho^2\beta).$$

This lemma is analogous to Lemma 4 in the stochastic case. The proof is very similar, with the slight difference that Lemma 5 is used instead of Lemma 3.

Proof. Let us define by $w_{t+1/2} = w_t + \rho\nabla L_{t+1/2}(w_t)$. Using the smoothness of the function L **(A2)**, we obtain

$$L(w_{t+1}) \leq L(w_t) - \gamma\langle \nabla L_{t+1}(w_{t+1/2}), \nabla L(w_t)\rangle + \frac{\gamma^2\beta}{2}\|\nabla L_{t+1}(w_{t+1/2})\|^2.$$

Taking the expectation and using that the variance is bounded **(A1)** yields to

$$\begin{aligned}\mathbb{E}\, L(w_{t+1}) &\leq \mathbb{E}\, L(w_t) - \gamma\, \mathbb{E}\langle \nabla L(w_{t+1/2}), \nabla L(w_t)\rangle + \frac{\gamma^2\beta}{2}\, \mathbb{E}\, \|\nabla L_{t+1}(w_{t+1/2})\|^2 \\ &\leq \mathbb{E}\, L(w_t) - \gamma\, \mathbb{E}\langle \nabla L(w_{t+1/2}), \nabla L(w_t)\rangle + \gamma^2\beta\, \mathbb{E}\, \|\nabla L_{t+1}(w_{t+1/2}) - \nabla L(w_{t+1/2})\|^2 \\ &\quad + \gamma^2\beta\, \mathbb{E}\, \|\nabla L(w_{t+1/2})\|^2 \\ &\leq \mathbb{E}\, L(w_t) - \gamma\, \mathbb{E}\langle \nabla L(w_{t+1/2}), \nabla L(w_t)\rangle + \gamma^2\beta\frac{\sigma^2}{b} + \gamma^2\beta\, \mathbb{E}\, \|\nabla L(w_{t+1/2})\|^2.\end{aligned}$$

The main trick is still to use the binomial squares

$$\|\nabla L(w_{t+1/2})\|^2 = -\|\nabla L(w_t)\|^2 + \|\nabla L(w_{t+1/2}) - \nabla L(w_t)\|^2 + 2\langle \nabla L(w_{t+1/2}), \nabla L(w_t)\rangle$$

to bound

$$\begin{aligned}\mathbb{E}\, L(w_{t+1}) &\leq \mathbb{E}\, L(w_t) - \gamma\, \mathbb{E}\langle \nabla L(w_{t+1/2}), \nabla L(w_t)\rangle + \frac{\gamma^2\beta}{2}\, \mathbb{E}\, \|\nabla L(w_{t+1/2})\|^2 + \gamma^2\sigma^2\beta/b \\ &= \mathbb{E}\, L(w_t) - \gamma^2 L\, \mathbb{E}\, \|\nabla L(w_t)\|^2 + \gamma^2\beta\, \mathbb{E}\, \|\nabla L(w_{t+1/2}) - \nabla L(w_t)\|^2 \\ &\quad - \gamma(1-2\gamma\beta)\, \mathbb{E}\langle \nabla L(w_{t+1/2}), \nabla L(w_t)\rangle + \gamma^2\sigma^2\beta/b \\ &= \mathbb{E}\, L(w_t) - \gamma^2\beta\, \mathbb{E}\, \|\nabla L(w_t)\|^2 + \gamma^2 L^3\, \mathbb{E}\, \|w_{t+1/2} - w_t\|^2 \\ &\quad - \gamma(1-2\gamma\beta)(1/2+\alpha\rho)\, \mathbb{E}\, \|\nabla L(w_t)\|^2 + \gamma(1-2\gamma L)\sigma^2\rho^2\beta^2/2 + \gamma^2\sigma^2\beta/b \\ &= \mathbb{E}\, L(w_t) - \gamma^2\beta\, \mathbb{E}\, \|\nabla L(w_t)\|^2 + \gamma^2\beta^3\rho^2\, \mathbb{E}\, \|\nabla L_{t+1/2}(w_t)\|^2 \\ &\quad - \gamma(1-2\gamma\beta)(1/2+\alpha\rho)\, \mathbb{E}\, \|\nabla L(w_t)\|^2 + \gamma(1-2\gamma\beta)\sigma^2/b\rho^2\beta^2/2 + \gamma^2\sigma^2\beta/b \\ &= \mathbb{E}\, L(w_t) - \gamma^2\beta\, \mathbb{E}\, \|\nabla L(w_t)\|^2 + 2\gamma^2\beta^3\rho^2\, \mathbb{E}\, \|\nabla L(w_t)\|^2 + 2\gamma^2\beta^3\rho^2\sigma^2/b \\ &\quad - \gamma(1-2\gamma\beta)(1/2+\alpha\rho)\, \mathbb{E}\, \|\nabla L(w_t)\|^2 + \gamma(1-2\gamma\beta)\sigma^2\rho^2\beta^2/2 + \gamma^2\sigma^2\beta/b \\ &\leq L(w_t) - \frac{\gamma}{2}[1 - 2\rho\beta(1-2\gamma\beta(1-\rho\beta))]\, \mathbb{E}\, \|\nabla L(w_t)\|^2 + \gamma\sigma^2\beta/b[\gamma + \rho^2 L/2(1+2\gamma\beta)]\end{aligned}$$

where we have used Lemma 5 and that $\|\nabla L(w_{t+1/2}) - \nabla L(w_t)\|^2 \leq \beta^2\|w_{t+1/2} - w_t\|^2$. □

Using Lemma 6 we directly obtain the following convergence result.

Theorem 5.10.6. *Assume **(A1)** and **(A2')**. For $\gamma \leq 1/(2\beta)$ and $\rho \leq 1/(2\beta)$, the iterates (5.4) satisfies:*

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\, \|\nabla L(w_t)\|^2 \leq 4\frac{L(w_0) - \mathbb{E}\, L(w_T)}{T\gamma} + 4T\sigma^2\beta(\gamma + \rho^2\beta)/b.$$

This theorem gives the first part of Theorem 5.10.5. The proof of the stronger result obtained when the function is in addition PL (Assumption **(A3)**) is similar to the proof of Theorem 3.2 of Gower et al. (2019), only the constants are changing.

Convergence of m-SAM

In the m-SAM algorithm, the *same* batch is used in the ascent and descent steps unlike in the n-SAM algorithm analyzed above. We obtain then iterates (5.4) for which we have stated the convergence result in Theorem 5.6.1 in the main part. The proof follows the same lines as above with the minor difference that we are assuming the *individual* gradients ∇f_t are Lipschitz (Assumption **(A2)**) to control the alignment of the expected SAM direction. Let us denote by $\nabla L_t(w) = \frac{1}{b}\sum_{i\in J_t}\nabla \ell_i(w)$.

Lemma 7. *Assume **(A1-2)**. Then we have for all $w \in \mathbb{R}^d$, $\rho \geq 0$ and $t \geq 0$*

$$\mathbb{E}\langle \nabla L_t(w+\rho\nabla L_t(w)), \nabla L(w)\rangle \geq (1/2 - \rho\beta)\|\nabla L(w)\|^2 - \frac{\beta^2\rho^2\sigma^2}{2b}.$$

The proof is very similar to the proof of Lemma 5. The only difference is that the Assumption **(A2)** is used instead of **(A2')**.

Proof. Let us denote by $\hat{w} = w + \rho\nabla L(w)$, the true gradient step. We first add and subtract $\nabla L_t(\hat{w})$

$$\langle \nabla L_t(w+\rho\nabla L_t(w)), \nabla L(w)\rangle = \langle \nabla L_t(w+\rho\nabla L_t(w)) - \nabla L_t(\hat{w}), \nabla L(w)\rangle - \langle \nabla L_t(\hat{w}), \nabla L(w)\rangle.$$

We bound the two terms separately. We use the smoothness of L_t to bound the first term (Assumption **(A2)**):

$$\begin{aligned}
-\langle \nabla L_t(w+\rho\nabla L_t(w)) - \nabla L_t(\hat{w}), \nabla L(w)\rangle &\leq \frac{1}{2}\|\nabla L_t(w+\rho\nabla L_t(w)) - \nabla L_t(\hat{w})\|^2 + \frac{1}{2}\|\nabla L(w)\|^2 \\
&\leq \frac{\beta^2}{2}\,\mathbb{E}\,\|w+\rho\nabla L_t(w) - \hat{w}\|^2 + \frac{1}{2}\|\nabla L(w)\|^2 \\
&\leq \frac{\beta^2\rho^2}{2}\|\nabla L_t(w) - \nabla L(w)\|^2 + \frac{1}{2}\|\nabla L(w)\|^2.
\end{aligned}$$

And taking the expectation, we obtain:

$$-\,\mathbb{E}\langle \nabla L_t(w+\rho\nabla L_t(w)) - \nabla L_t(\hat{w}), \nabla L(w)\rangle \leq \frac{\beta^2\rho^2\sigma^2}{2b} + \frac{1}{2}\,\mathbb{E}\,\|\nabla L(w)\|^2.$$

For the second term, we apply directly Lemma 3

$$\mathbb{E}\langle \nabla L_t(\hat{w}), \nabla L(w_t)\rangle = \langle \nabla L(\hat{w}), \nabla L(w)\rangle \geq (1-\beta\rho)\|\nabla L(w)\|^2.$$

Assembling the two inequalities yields the result. □

The next lemma shows that the decrease of function values of the m-SAM algorithm can be controlled similarly as in the case of gradient descent. It is analogous to Lemma 6 where *different* batches are used in both the ascent and descent steps of SAM algorithm.

Lemma 8. *Assume **(A1-2)**. For all $\gamma \leq \frac{1}{\beta}$ and $\rho \leq \frac{1}{4\beta}$, the iterates (5.4) satisfy*

$$\mathbb{E}\, L(w_{t+1}) \leq \mathbb{E}\, L(w_t) - \frac{3\gamma}{8}\, \mathbb{E}\, \|\nabla L(w_t)\|^2 + \gamma\beta \frac{\sigma^2}{b}(\gamma + 2\rho^2\beta).$$

Proof. Let us define by $w_{t+1/2} = w_t + \rho \nabla L_{t+1}(w_t)$. Using the smoothness of the function L which is implied by **(A2)**, we obtain

$$L(w_{t+1}) \leq L(w_t) - \gamma \langle \nabla L_{t+1}(w_{t+1/2}), \nabla L(w_t) \rangle + \frac{\gamma^2\beta}{2} \|\nabla L_{t+1}(w_{t+1/2})\|^2.$$

We still use the binomial squares

$$\|\nabla L_{t+1}(w_{t+1/2})\|^2 = -\|\nabla L(w_t)\|^2 + \|\nabla L_{t+1}(w_{t+1/2}) - \nabla L(w_t)\|^2 + 2\langle \nabla L_{t+1}(w_{t+1/2}), \nabla L(w_t) \rangle$$

and bound $L(w_{t+1})$ by

$$\begin{aligned}
L(w_{t+1}) &\leq L(w_t) - \frac{\gamma^2\beta}{2}\|\nabla L(w_t)\|^2 + \frac{\gamma^2\beta}{2}\|\nabla L_{t+1}(w_{t+1/2}) - \nabla L(w_t)\|^2 - \gamma(1-\gamma\beta)\langle \nabla L_{t+1}(w_{t+1/2}), \nabla L(w_t) \rangle \\
&\leq L(w_t) - \frac{\gamma^2\beta}{2}\|\nabla L(w_t)\|^2 + \gamma^2\beta\|\nabla L_{t+1}(w_{t+1/2}) - \nabla L_{t+1}(w_t)\|^2 + \gamma^2\beta\|\nabla L_{t+1}(w_t) - \nabla L(w_t)\|^2 \\
&\quad - \gamma(1-\gamma\beta)\langle \nabla L_{t+1}(w_{t+1/2}), \nabla L(w_t) \rangle \\
&\leq L(w_t) - \frac{\gamma^2\beta}{2}\|\nabla L(w_t)\|^2 + \gamma^2\beta\beta^2\|w_{t+1/2} - w_t\|^2 + \gamma^2\beta\|\nabla L_{t+1}(w_t) - \nabla L(w_t)\|^2 \\
&\quad - \gamma(1-\gamma\beta)\langle \nabla L_{t+1}(w_{t+1/2}), \nabla L(w_t) \rangle \\
&= L(w_t) - \frac{\gamma^2\beta}{2}\|\nabla L(w_t)\|^2 + \gamma^2\beta^3\rho^2\|\nabla L_{t+1}(w_t)\|^2 + \gamma^2\beta\|\nabla L_{t+1}(w_t) - \nabla L(w_t)\|^2 \\
&\quad - \gamma(1-\gamma\beta)\langle \nabla L_{t+1}(w_{t+1/2}), \nabla L(w_t) \rangle \\
&= L(w_t) - \frac{\gamma^2\beta}{2}(1 - 4\beta^2\rho^2)\|\nabla L(w_t)\|^2 + \gamma^2\beta(1 + 2\beta^2\rho^2)\|\nabla L_{t+1}(w_t) - \nabla L(w_t)\|^2 \\
&\quad - \gamma(1-\gamma\beta)\langle \nabla L_{t+1}(w_{t+1/2}), \nabla L(w_t) \rangle
\end{aligned}$$

Taking the expectation and using Lemma 7, we obtain

$$\begin{aligned}
\mathbb{E}\, L(w_{t+1}) &\leq \mathbb{E}\, L(w_t) - \frac{\gamma^2\beta}{2}(1 - 4\beta^2\rho^2)\, \mathbb{E}\, \|\nabla L(w_t)\|^2 + \gamma^2\beta(1 + 2\beta^2\rho^2)\, \mathbb{E}\, \|\nabla L_{t+1}(w_t) - \nabla L(w_t)\|^2 \\
&\quad - \gamma(1-\gamma\beta)\, \mathbb{E}\langle \nabla L_{t+1}(w_{t+1/2}), \nabla L(w_t) \rangle \\
&\leq \mathbb{E}\, L(w_t) - \frac{\gamma^2\beta}{2}(1 - 4\beta^2\rho^2)\, \mathbb{E}\, \|\nabla L(w_t)\|^2 + \gamma^2\beta(1 + 2\beta^2\rho^2)\sigma^2/b \\
&\quad - \gamma(1-\gamma\beta)(1/2 - \beta\rho)\, \mathbb{E}\, \|\nabla L(w_t)\|^2 + \gamma(1-\gamma\beta)\frac{\rho^2\sigma^2\beta^2}{2b} \\
&\leq \mathbb{E}\, L(w_t) - \frac{\gamma^2\beta}{2}(1 - 4\beta^2\rho^2)\, \mathbb{E}\, \|\nabla L(w_t)\|^2 + \gamma^2\beta(1 + 2\beta^2\rho^2)\sigma^2/b
\end{aligned}$$

$$-\frac{\gamma}{2}(1-2\beta\rho(1-\gamma(\beta-2\rho\beta^2)))\,\mathbb{E}\,\|\nabla L(w_t)\|^2+\gamma\sigma^2/b[\gamma\beta+\frac{\rho^2\beta^2}{2}(1+3\gamma\beta)].$$

□

Using Lemma 8 we directly obtain the main convergence result for m-SAM.

Theorem 5.10.7. *Assume **(A1-2)**. For $\gamma \leq \frac{1}{\beta}$ and $\rho \leq \frac{1}{4\beta}$, the iterates (5.4) satisfy:*

$$\frac{1}{T}\,\mathbb{E}\left[\sum_{t=0}^{T-1}\|\nabla L(w_t)\|^2\right] \leq \frac{8}{3T\gamma}\left(L(w_0)-\mathbb{E}\,L(w_T)\right)+\frac{8\sigma^2\beta(\gamma+\rho^2\beta)}{3b}.$$

*In addition, under **(A3)**, with step sizes $\gamma_t = \min\{\frac{8t+4}{3\mu(t+1)^2}, \frac{1}{2\beta}\}$ and $\rho_t = \sqrt{\gamma_t/\beta}$:*

$$\mathbb{E}[L(w_T)]-L_* \leq \frac{3\beta^2(L(w_0)-L_*)}{\mu^2T^2}+\frac{22\beta\sigma^2}{\mu^2bT}.$$

Proof. The first bound directly comes from Lemma 8. The second bound is similar to the proof of Theorem 3.2 of Gower et al. (2019), only the constants are changing. □

Finally, we note that Theorem 5.6.1 is a direct consequence of Theorem 5.10.7 with $\gamma_t = \frac{1}{\sqrt{T}\beta}$, $\rho_t = \frac{1}{T^{1/4}\beta}$ and slightly simplified constants.

5.11 Experimental Details

Training details for deep networks. In all experiments, we train deep networks using SGD with step size 0.1, momentum 0.9, and ℓ_2-regularization parameter $\lambda = 0.0005$. We perform experiments on CIFAR-10 and CIFAR-100 (Krizhevsky and Hinton, 2009) where for all experiments we apply basic data augmentations: random image crops and mirroring. We use batch size 128 for most experiments except when it is mentioned otherwise. We use a pre-activation ResNet-18 (He et al., 2016b) for CIFAR-10 and ResNet-34 on CIFAR-100 with a width factor 64 and piece-wise constant learning rates (with a 10-times decay at 50% and 75% epochs). We train all models for 200 epochs except those in Sec. 5.5.3 and Sec. 5.6.2 for which we use 1000 epochs. We use batch normalization for most experiments, except when it is explicitly mentioned otherwise as, for example, in the experiments where we aim to compute sharpness and for this we use networks with group normalization.

For all experiments involving SAM, we select the best perturbation radius ρ based on a grid search over $\rho \in \{0.025, 0.05, 0.1, 0.2, 0.3, 0.4\}$. In most cases, the optimal ρ is equal to 0.1 while in the ERM $\rightarrow$ SAM experiment, it is equal to $\rho = 0.4$ for CIFAR-10 and $\rho = 0.2$ for CIFAR-100. We note that using a higher ρ in this case is coherent with the experiments on diagonal linear networks which also required a higher ρ. For all experiments with SAM, we use a single GPU, so we do not implicitly rely on lower m-sharpness in m-SAM. The only exception where m is smaller than the batch size is the experiments shown in Fig. 5.4

and Fig. 5.16. Regarding n-SAM in Fig. 5.1, we implement it by doing the ascent step on a *different* batch compared to the descent step, i.e., as described in our convergence analysis part in Eq. (5.21).

Sharpness computation. We compute m-sharpness on 1024 training points (i.e., by averaging over $\lceil 1024/m \rceil$) of CIFAR-10 or CIFAR-100 using 100 iterations of projected gradient ascent using a step size $\alpha = 0.1 \cdot \rho$. For each iteration, we normalize the updates by the ℓ_2 gradient norm.

Confidence intervals on plots. Many experimental results are replicated over different random seeds used for training. We show the results using the mean and 95% bootstrap confidence intervals which is the standard way to show such results in the `seaborn` library Waskom (2021).

Code and computing infrastructure. The code of our experiments is publicly available.[4] We perform all our experiments with deep networks on a single NVIDIA V100 GPU with 32GB of memory. Since most of our experiments involved a grid search over the perturbation radius ρ and replication over multiple random seeds, we could not do the same at the ImageNet scale due to our limited computational resources.

5.12 Additional Deep Learning Experiments

In this section, we show additional experimental results complementary to those presented in the main part. In particular, we provide multiple ablation study related to the role of m in m-SAM, batch size, and model width. We also provide additional experiments on the evolution of sharpness over training using training time and test time batch normalization, training loss of ERM vs. SAM models, and the performance under label noise for standard and unnormalized SAM.

5.12.1 The Effect of m in m-SAM

We show the results of SAM for different m in m-SAM (with a fixed batch size 256) in Fig. 5.16. We note that in this experiment, we used group normalization instead of batch normalization like, for example, in Fig. 5.1, so the exact test error values should not be compared between these two figures. We observe from Fig. 5.16, that the generalization improvement is larger for smaller m and it is continuous in m. We also note that a similar experiment has been done in the original SAM paper (Foret et al., 2021). Here, we additionally verified this finding on an additional dataset (CIFAR-100) and for networks trained without batch normalization (which may have had an extra regularization effect as we discussed in Sec. 5.5.1).

[4] https://github.com/tml-epfl/understanding-sam

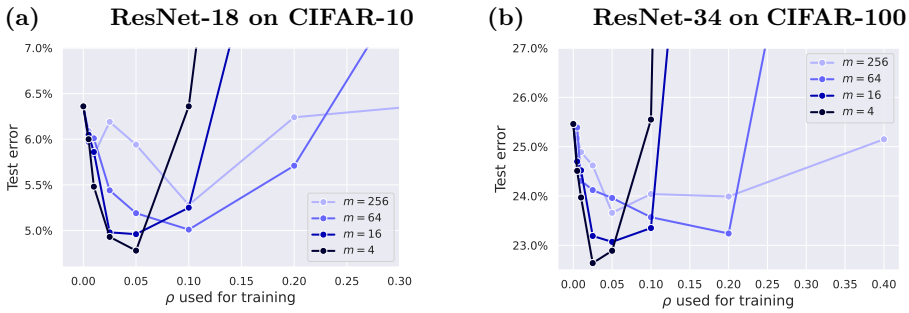


Figure 5.16: Test error of models trained with group normalization and **different m** in m-SAM using batch size 256.

5.12.2 The Effect of the Batch Size on SAM

We show the results of SAM for different batch sizes in Fig. 5.17 where we use m equal to the batch size. Note that a too high m leads to marginal improvements in generalization ($\approx 0.2\%$) and is not able to bridge the gap between large-batch (1024) and small-batch (256 or 128) SGD.

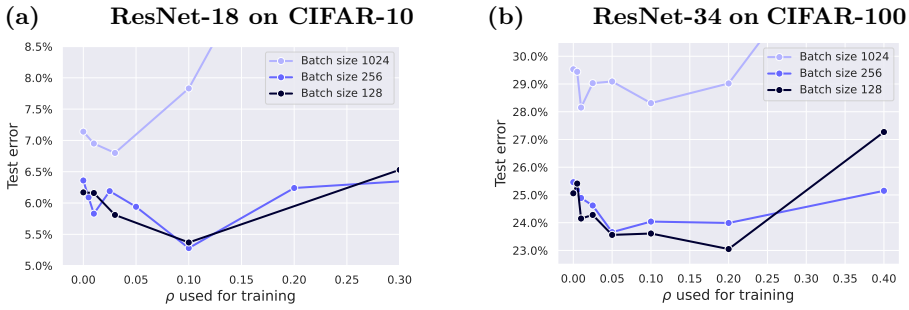


Figure 5.17: Test error of models trained with group normalization and **different batch sizes** for the same number of epochs (200). Note that for all models, we use m in m-SAM equal to the batch size.

5.12.3 The Effect of the Model Width on SAM

We show in Fig. 5.18 test error improvements of SAM over ERM for different model width factors. For comparison, in all other experiments we use model width factor 64. As expected, there is little improvement (or even no improvement as on CIFAR-10) from SAM for small networks where extra regularization is not needed. However, interestingly, the generalization improvement is the largest not for the widest models, but rather for intermediate model widths, such as model width 16.

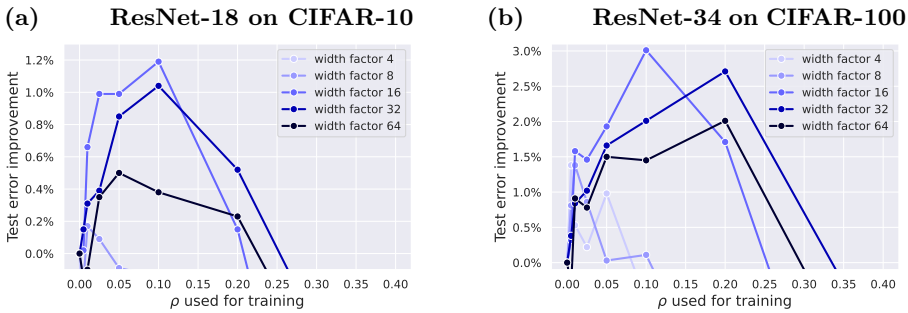


Figure 5.18: Test error improvements of SAM over ERM for **different model width** factors.

5.12.4 Sharpness for Models with Batch Normalization

The main problem of measuring sharpness for networks with BatchNorm is the discrepancy between training and test-time behaviour. Fig. 5.19 illustrates this issue: the maximum loss computed over radius ρ is substantially different depending on whether we use training-time vs. test-time BatchNorm. This is an important discrepancy since the training-time BatchNorm is effectively used by SAM while the test-time BatchNorm is used by default for post-hoc sharpness computation. To avoid this discrepancy, we presented the results in the main part only on models trained with GroupNorm which does not have this problem.

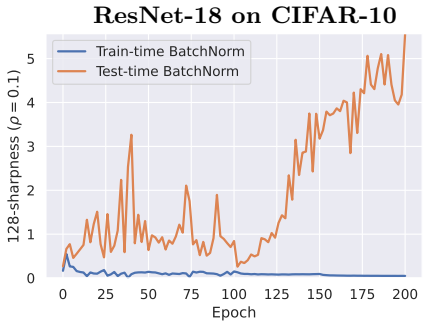


Figure 5.19: 128-sharpness ($\rho = 0.1$) over training for a network with batch normalization when measured with the training-time and test-time batch normalization. The model is trained with SAM using $\rho = 0.1$.

5.12.5 Training Loss for ERM vs. SAM Models

Fig. 5.11 in the main part shows that both training and test errors have a slight increasing trend after the first learning rate decay at 500 epochs. As a sanity check, in Fig. 5.20, we plot the total objective value (including the ℓ_2 regularization term) which shows a consistent decreasing trend. Thus, we conclude that the increasing training error is not some anomaly connected to a failure of optimizing the training objective.

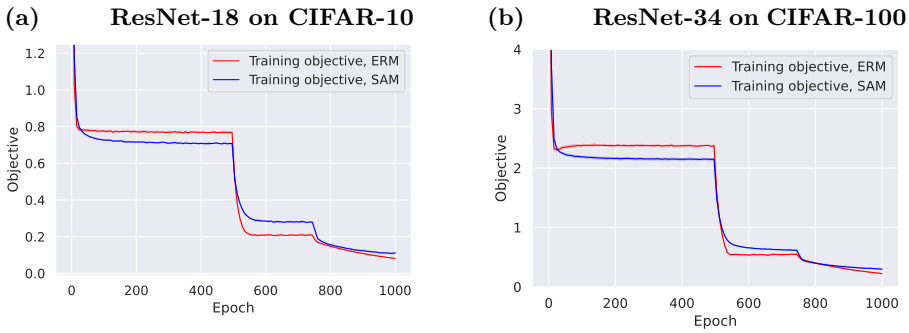


Figure 5.20: Training objective of ERM vs. SAM over epochs. For both models, we observe a clear decreasing trend.

5.12.6 SAM with a Decreasing Perturbation Radius

In Fig. 5.21, we plot the test error over different ρ_t where we decay the ρ_t using the same schedule as for the outer learning rate γ_t. We denote this as *SAM with decreasing* ρ contrary to the standard SAM for which ρ is constant throughout training. We note that in both cases, we use the ℓ_2-normalized updates as in the original SAM. The results suggest that decreasing the perturbation radius ρ_t over epochs is detrimental to generalization. This observation is relevant in the context of the convergence analysis that suggests that SAM converges even if ρ_t is significantly larger than the outer step size γ_t which is the case when we decay γ_t over epochs while keeping ρ_t constant.

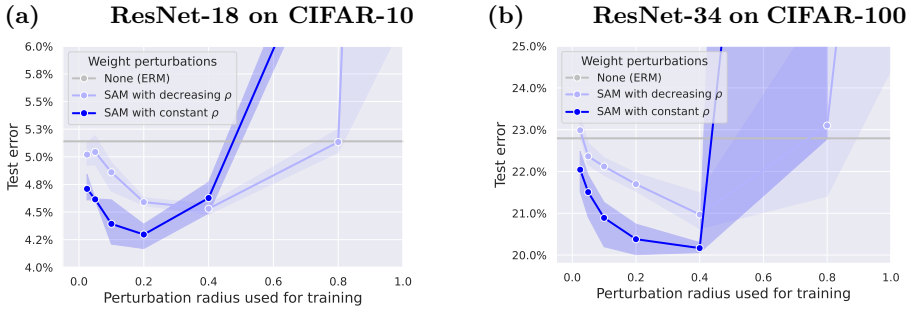


Figure 5.21: Test error of SAM with a constant perturbation radius ρ (i.e., standard SAM) compared to SAM with decreasing perturbation radii ρ_t. The decrease of ρ_t follows the same piecewise constant schedule as the learning rate γ_t. We note that in both cases, we use the ℓ_2-normalized updates as in the original SAM.

5.12.7 Experiments with Noisy Labels

In Fig. 5.22, we show experiments with CIFAR-10 and CIFAR-100 with 60% of noisy labels for SAM with a fixed inner step size ρ that does not include gradient normalization (denoted as *unnormalized SAM*). We did a prior grid search to determine the best fixed ρ for this case which we show in the figure. We can observe that the best test error taken over epochs almost exactly matches that of the standard SAM.

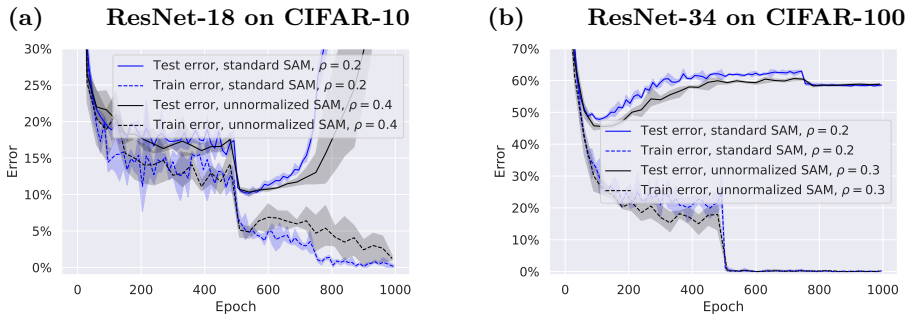


Figure 5.22: Plots over training for a ResNet-18 trained on CIFAR-10 with 60% label noise for SAM with and without gradient normalization.

6 SGD with Large Step Sizes Learns Sparse Features

6.1 Preface

In this chapter, based on Andriushchenko et al. (2023d) we present a study of the dynamics of the Stochastic Gradient Descent (SGD) in the training of neural networks. We show that commonly used large step sizes may lead the iterates to jump from one side of a valley to the other causing *loss stabilization*, and this stabilization induces a hidden stochastic dynamics that *biases it implicitly* toward sparse predictors.

Summary We showcase important features of the dynamics of the Stochastic Gradient Descent (SGD) in the training of neural networks. We present empirical observations that commonly used large step sizes (i) may lead the iterates to jump from one side of a valley to the other causing *loss stabilization*, and (ii) this stabilization induces a hidden stochastic dynamics that *biases it implicitly* toward *sparse* predictors. Furthermore, we show empirically that the longer large step sizes keep SGD high in the loss landscape valleys, the better the implicit regularization can operate and find sparse representations. Notably, no explicit regularization is used: the regularization effect comes solely from the SGD dynamics influenced by the large step sizes schedule. Therefore, these observations unveil how, through the step size schedules, both gradient and noise drive together the SGD dynamics through the loss landscape of neural networks. We justify these findings theoretically through the study of simple neural network models as well as qualitative arguments inspired from stochastic processes. This analysis allows us to shed new light on some common practices and observed phenomena when training deep networks. The code of our paper is available at https://github.com/tml-epfl/sgd-sparse-features.

Co-authors Aditya Varre, Loucas Pillaud-Vivien, Nicolas Flammarion.

Contributions Maksym Andriushchenko proposed the project and performed all experiments. Aditya Varre performed the theoretical analysis.

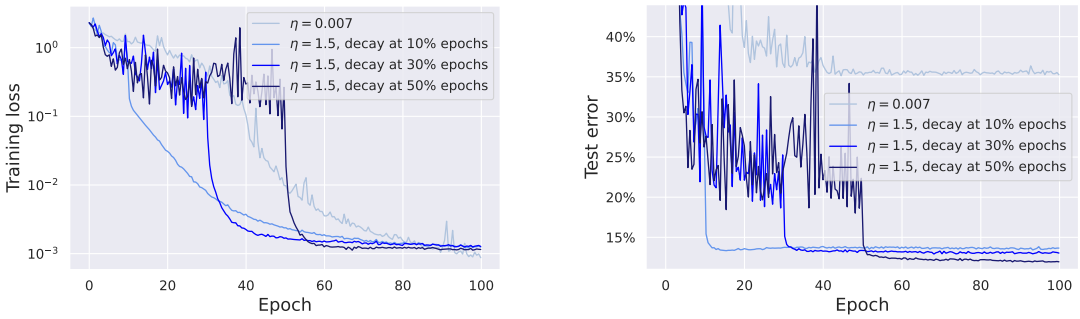


Figure 6.1: A typical training dynamics for a ResNet-18 trained on CIFAR-10. We use weight decay but no momentum or data augmentation for this experiment. We see a substantial difference in generalization (as large as 12% vs. 35% test error) depending on the step size η and its schedule. When the training loss stabilizes, there is a hidden progress occurring which we aim to characterize.

6.2 Introduction

Deep neural networks have accomplished remarkable achievements on a wide variety of tasks. Yet, the understanding of their remarkable effectiveness remains incomplete. From an optimization perspective, stochastic training procedures challenge many insights drawn from convex models. Notably, large step size schedules used in practice lead to unexpected patterns of stabilizations and sudden drops in the training loss (He et al., 2016a). From a generalization perspective, overparametrized deep nets generalize well while fitting perfectly the data and without any explicit regularizers (Zhang et al., 2017a). This suggests that optimization and generalization are tightly intertwined: neural networks find solutions that generalize well *thanks* to the optimization procedure used to train them. This property, known as *implicit bias* or *algorithmic regularization*, has been studied both for regression (Li et al., 2018; Woodworth et al., 2020) and classification (Soudry et al., 2018; Lyu and Li, 2020; Chizat and Bach, 2020a). However, for these theoretical results, it is also shown that typical timescales needed to enter the beneficial feature learning regimes are prohibitively long (Woodworth et al., 2020; Moroshko et al., 2020).

In this paper, we aim at staying closer to the experimental practice and consider the SGD schedules from the ResNet paper (He et al., 2016a) where the *large step size* is first kept constant and then decayed, potentially multiple times. We illustrate this behavior in Fig. 6.1 where we reproduce a minimal setting without data augmentation or momentum, and with only one step size decrease. We draw attention to two key observations regarding the large step size phase: (a) quickly after the start of training, the loss remains approximately constant on average and (b) despite no progress on the training loss, running this phase for longer leads to better generalization. We refer to such large step size phase as *loss stabilization*. The better generalization hints at some *hidden dynamics* in the parameter space not captured by the loss curves in Fig. 6.1. Our main contribution is to unveil the hidden dynamics behind this phase: loss stabilization helps to amplify the noise of SGD that drives the network towards a solution with *sparser features*, meaning that for a feature vector $\psi(x)$, only a few unique features are active for a given input x.

6.2.1 Our Contributions

The effective dynamics behind loss stabilization. We characterize two main components of the SGD dynamics with large step sizes: (i) a fast movement determined by the bouncing directions causing loss stabilization, (ii) a slow dynamics driven by the combination of the gradient and the multiplicative noise—which is non-vanishing due to the loss stabilization.

SDE model and sparse feature learning. We model the *effective* slow dynamics during loss stabilization by a stochastic differential equation (SDE) whose multiplicative noise is related to the neural tangent kernel features, and validate this modeling experimentally. Building on the existing theory on diagonal linear networks, which shows that this noise structure leads to sparse predictors, we conjecture a similar "sparsifying" effect on the features of more complex architectures. We experimentally confirm this on neural networks of increasing complexity.

Insights from our understanding. We draw a clear general picture: the hidden optimization dynamics induced by large step sizes and loss stabilization enable the transition to a sparse feature learning regime. We argue that after a short initial phase of training, SGD *first* identifies sparse features of the training data and eventually fits the data when the step size is decreased. Finally, we discuss informally how many deep learning regularization methods (weight decay, BatchNorm, SAM) may also fit into the same picture.

6.2.2 Related Work

He et al. (2016a) popularized the piece-wise constant step size schedule which often exhibits a clear loss stabilization pattern which was later characterized theoretically in Li et al. (2020b) from the optimization point of view. However, the regularization effect of this phase induced by the underlying *hidden stochastic dynamics* is still unclear. Li et al. (2019d) analyzed the role of loss stabilization for a synthetic distribution containing different patterns, but it is not clear how this analysis can be extended to general problems. Jastrzebski et al. (2021) suggest that large step sizes prevent the increase of local curvature during the early phase of training. However, they do not provide an explanation for this phenomenon.

The importance of large step sizes for generalization has been investigated with diverse motivations. Many works conjectured that large step sizes induce minimization of some complexity measures related to the flatness of minima (Keskar et al., 2016; Smith and Le, 2018; Smith et al., 2021; Yang et al., 2022). Notably, Xing et al. (2018) point out that SGD moves through the loss landscape bouncing between the walls of a valley where the role of large step sizes is to guide the SGD iterates towards a flatter minimum. However, the correct flatness measure is often disputed (Dinh et al., 2017) and its role in understanding generalization is questionable since full-batch GD with large step sizes (unlike SGD) can lead to flat solutions which don't generalize well (Kaur et al., 2022)

The attempts to explain the effect of large step size on strongly convex models (Nakkiran, 2020; Wu et al., 2021; Beugnot et al., 2022) are inherently incomplete since it is a phenomenon related to the existence of many zero solutions with very different generalization properties. Works based on stability analysis characterize the properties of the minimum that SGD or GD can potentially converge depending on the step size (Wu et al., 2018; Mulayoff et al., 2021; Ma and Ying, 2021; Nacson et al., 2022). However, these approaches *do not capture the entire training dynamics* such as the large step size phase that we consider where SGD converges only after the step size is decayed.

To grasp the generalization of SGD, research has focused on SGD augmented with label noise due to its beneficial regularization properties and resemblance to the standard noise of SGD. Its implicit bias has been first characterized by Blanc et al. (2020) and extended by Li et al. (2022). However, their analysis only holds in the final phase of the training, close to a zero-loss manifold. Our work instead is closer in spirit to Pillaud-Vivien et al. (2022) where the label noise dynamics is analyzed in the *central* phase of the training, i.e., when the loss is still substantially above zero.

The dynamics of GD with large step sizes have received a lot of attention in recent times, particularly the edge-of-stability phenomenon Cohen et al. (2021) and the catapult mechanism (Lewkowycz et al., 2020; Wang et al., 2022). However, the lack of stochastic noise in their analysis renders them incapable of capturing stochastic training. Note that it is possible to bridge the gap between GD and SGD by using explicit regularization as in Geiping et al. (2022). We instead focus on the *implicit* regularization of SGD which remains the most practical approach for training deep nets.

Finally, sparse features and low-rank structures in deep networks have been commonly used for model compression, knowledge distillation, and lottery ticket hypothesis (Denton et al., 2014; Hinton et al., 2015; Frankle and Carbin, 2018). A common theme of all these works is the presence of *hidden structure* in the networks learned by SGD which allows one to come up with a much smaller network that approximates well the original one. In particular, Hoefler et al. (2021) note that ReLU activations in deep networks trained with SGD are typically much sparser than 50%. Our findings suggest that the step size schedule can be the key component behind emergence of such sparsity.

6.3 The Effective Dynamics of Large Step Size SGD: Sparse Feature Learning

In this section, we show that large step sizes may lead the loss to stabilize by making SGD bounce above a valley. We then unveil the effective dynamics induced by this loss stabilization. To clarify our exposition we showcase our results for the mean square error but other losses like the cross-entropy carry the same key properties in terms of the noise covariance (Wojtowytsch, 2021b, Lemma 2.14). We consider a generic parameterized family of prediction functions $\mathcal{H} := \{x \to h_\theta(x),\ \theta \in \mathbb{R}^p\}$, a setting which encompasses

neural networks. In this case, the training loss on input/output samples $(x_i, y_i)_{1\leq i\leq n} \in \mathbb{R}^d \times \mathbb{R}$ is equal to

$$\mathcal{L}(\theta) := \frac{1}{2n}\sum_{i=1}^{n} \left(h_\theta(x_i) - y_i\right)^2 . \tag{6.1}$$

We consider the overparameterized setting, i.e. $p \gg n$, hence, there shall exists many parameters θ^* that lead to zero loss, i.e., perfectly interpolate the dataset. Therefore, the question of which interpolator the algorithm converges to is of paramount importance in terms of generalization. We focus on the SGD recursion with step size $\eta > 0$, initialized at $\theta_0 \in \mathbb{R}^p$: for all $t \in \mathbb{N}$,

$$\theta_{t+1} = \theta_t - \eta(h_{\theta_t}(x_{i_t}) - y_{i_t})\nabla_\theta h_{\theta_t}(x_{i_t}), \tag{6.2}$$

where $i_t \sim \mathcal{U}\left(\llbracket 1, n \rrbracket\right)$ is the uniform distribution over the sample indices. In the following, note that SGD with mini batches of size $B > 1$ would lead to similar analysis but with η/B instead of η.

6.3.1 Background: SGD is GD with Specific Label Noise

To emphasize the combined roles of gradient and noise, we highlight the connection between the SGD dynamics and that of full-batch GD plus a specific label noise. Such manner of reformulating the dynamics has already been used in previous works attempting to understand the specificity of the SGD noise (HaoChen et al., 2020; Ziyin et al., 2022). We formalize it in the following proposition.

Proposition 6.3.1. *Let $(\theta_t)_{t\geq 0}$ follow the SGD dynamics (6.2) with the random sampling function $(i_t)_{t\geq 0}$. For $t \geq 0$, define the random vector $\xi_t \in \mathbb{R}^n$ such that*

$$[\xi_t]_i := (h_{\theta_t}(x_i) - y_i)(1 - n\mathbf{1}_{i=i_t}), \tag{6.3}$$

for $i \in \llbracket 1, n \rrbracket$ and where $\mathbf{1}_A$ is the indicator of the event A. Then $(\theta_t)_{t\geq 0}$ follows the full-batch gradient dynamics on $\mathcal{L}$ with label noise $(\xi_t)_{t\geq 0}$, that is

$$\theta_{t+1} = \theta_t - \frac{\eta}{n}\sum_{i=1}^{n}(h_{\theta_t}(x_i) - y_i^t)\nabla_\theta h_{\theta_t}(x_i), \tag{6.4}$$

where we define the random labels $y^t := y + \xi_t$. Furthermore, ξ_t is a mean zero random vector with variance such that $\frac{1}{n(n-1)}\mathbb{E}\left\|\xi_t\right\|^2 = 2\mathcal{L}(\theta_t)$.

This reformulation shows two crucial aspects of the SGD noise: (i) the noisy part at state θ always belongs to the linear space spanned by $\{\nabla_\theta h_\theta(x_1), \ldots, \nabla_\theta h_\theta(x_n)\}$, and (ii) it scales as the training loss. Going further on (ii), we highlight in the following section that the loss can stabilize because of large step sizes: this may lead to a *constant* effective scale of label noise. These two features are of paramount importance when modelling the effective dynamics that take place during loss stabilization.

6.3.2 The Effective Dynamics Behind Loss Stabilization

On loss stabilization. For generic quadratic costs, e.g., $F(\beta) := \|X\beta - y\|^2$, gradient descent with step size η is convergent for $\eta < 2/\lambda_{\max}$, divergent for $\eta > 2/\lambda_{\max}$ and converges to a bouncing 2-periodic dynamics for $\eta = 2/\lambda_{\max}$, where $\lambda_{\max}$ is the largest eigenvalue of the Hessian. However, the practitioner is not likely to hit perfectly this unstable step size and, almost surely, the dynamics shall either converge or diverge. Yet, non-quadratic costs bring to this picture a particular complexity: it has been shown that, even for non-convex toy models, there exist an open interval of step sizes for which the gradient descent neither converge nor diverge (Ma et al., 2022; Chen and Bruna, 2022). As we are interested in SGD, we complement this result by presenting an example in which loss stabilization occurs almost surely *in the case of stochastic updates*. Indeed, consider a regression problem with quadratic parameterization on one-dimensional data inputs x_i's, coming from a distribution $\hat{\rho}$, and outputs generated by the linear model $y_i = x_i\theta_*^2$. The loss writes $F(\theta) := \frac{1}{4}\mathbb{E}_{\hat{\rho}}\left(y - x\theta^2\right)^2$, and the SGD iterates with step size $\eta > 0$ follow, for any $t \in \mathbb{N}$,

$$\theta_{t+1} = \theta_t + \eta\,\theta_t\,x_{i_t}\left(y_{i_t} - x_{i_t}\theta_t^2\right) \text{ where } \quad x_{i_t} \sim \hat{\rho}. \tag{6.5}$$

For the sake of clarity, suppose that $\theta_* = 1$ and $\mathrm{supp}(\hat{\rho}) = [a, b]$, we have the following proposition (a more general result is presented in Proposition 6.7.1 of the Appendix).

Proposition 6.3.2. *For any $\eta \in (a^{-2}, 1.25 \cdot b^{-2})$ and initialization $\theta_0 \in (0, 1)$, for all $t > 0$,*

$$\delta_1 < F(\theta_t) < \delta_2 \text{ almost surely, and} \tag{6.6}$$

$$\exists T > 0, \forall k > T,\ \theta_{t+2k} < 1 < \theta_{t+2k+1} \text{ almost surely.} \tag{6.7}$$

where $\delta_1, \delta_2, T > 0$ are constant given in the Appendix.

The proposition is divided in two parts: if the step size is large enough, (6.6) the loss stabilizes in between level sets δ_1 and δ_2 and (6.7) shows that after some initial phase, the iterates bounce from one side of the *loss valley* to the other one. Note that despite the stochasticity of the process, the results hold *almost surely*.

The effective dynamics. As observed in the prototypical SGD training dynamics of Fig. 6.1 and proved in the non-convex toy model of Proposition 6.3.2, large step sizes lead the loss to stabilize around some level set. To further understand the effect of this loss stabilization in parameter space, we shall assume perfect stabilization. Then, from Proposition 6.3.1, we conjecture the following behaviour

During loss stabilization, SGD is well modelled by GD with constant label noise.

Label noise dynamics have been studied recently (Blanc et al., 2020; Damian et al., 2021; Li et al., 2022) thanks to their connection with Stochastic Differential Equations (SDEs).

To properly write a SDE model, the drift should match the gradient descent and the noise should have the correct covariance structure (Li et al., 2019a; Wojtowytsch, 2021a). Proposition 6.3.1 implies that the noise at state θ is spanned by the gradient vectors $\{\nabla_\theta h_\theta(x_1), \dots, \nabla_\theta h_\theta(x_n)\}$ and has a constant intensity corresponding to the loss stabilization at a level $\delta > 0$. Hence, we propose the following SDE model

$$\mathrm{d}\theta_t = -\nabla_\theta \mathcal{L}(\theta_t)\mathrm{d}t + \sqrt{\eta\delta}\,\phi_{\theta_t}(X)^\top \mathrm{d}B_t, \tag{6.8}$$

where $(B_t)_{t\geq 0}$ is a standard Brownian motion in $\mathbb{R}^n$ and $\phi_\theta(X) := [\nabla_\theta h_\theta(x_i)^\top]_{i=1}^n \in \mathbb{R}^{n\times p}$ referred to as the *Jacobian* (which is also the *Neural Tangent Kernel (NTK) feature matrix* (Jacot et al., 2018)). This SDE can be seen as *the effective slow dynamics* that drives the iterates while they bounce *rapidly* in some directions at the level set δ. It highlights the combination of the deterministic part of the full-batch gradient and the noise induced by SGD. Beyond the theoretical justification and consistency of this SDE model, we validate it empirically in Sec. 6.8 showing that it indeed captures the dynamics of large step size SGD. In the next section, we leverage the SDE (6.8) to understand the implicit bias of such learning dynamics.

6.3.3 Sparse Feature Learning

We begin with a simple model of diagonal linear networks that showcase a sparsity inducing dynamics and further disclose our general message about the overall implicit bias promoted by the effective dynamics.

A warm-up: diagonal linear networks

An appealing example of simple non-linear networks that help in forging an intuition for more complicated architectures is diagonal linear networks (Vaskevicius et al., 2019; Woodworth et al., 2020; HaoChen et al., 2020; Pesme et al., 2021). They are two-layer linear networks with only diagonal connections: the prediction function writes $h_{u,v}(x) = \langle u, v \odot x\rangle = \langle u \odot v, x\rangle$ where $\odot$ denotes *elementwise* multiplication. Even though the loss is convex in the associated linear predictor $\beta := u \odot v \in \mathbb{R}^d$, it is not in (u, v), hence the training of such simple models already exhibit a rich non-convex dynamics. In this case, $\nabla_u h_{u,v}(x) = v \odot x$, and the SDE model (6.8) writes

$$\mathrm{d}u_t = -\nabla_u \mathcal{L}(u_t, v_t)\,\mathrm{d}t \;+ \sqrt{\eta\delta}\, v_t \odot \left[X^\top \mathrm{d}B_t\right], \tag{6.9}$$

where $(B_t)_{t\geq 0}$ is a standard Brownian motion in $\mathbb{R}^n$. Equations are symmetric for $(v_t)_{t\geq 0}$.

What is the behaviour of this effective dynamics? Pillaud-Vivien et al. (2022) answered this question by analyzing a similar stochastic dynamics and unveiled the sparse nature of the resulting solutions. Indeed, under sparse recovery assumptions, denoting β^* the sparsest linear predictor that interpolates the data, it is shown that the associated linear predictor $\beta_t = u_t \odot v_t$: (i) converges exponentially fast to zero outside of the support

of β^* (ii) is *with high probability* in a $\mathcal{O}(\sqrt{\eta\delta})$ neighborhood of β^* in its support after a time $\mathcal{O}(\delta^{-1})$.

Overall conclusion on the model. During a first phase, SGD with large step sizes η decreases the training loss until stabilization at some level set $\delta > 0$. During this loss stabilization, an effective noise-driven dynamics takes place. It shrinks the coordinates outside of the support of the sparsest signal and oscillates in parameter space at level $\mathcal{O}(\sqrt{\eta\delta})$ on its support. Hence, decreasing later the step size leads to perfect recovery of the sparsest predictor. This behaviour is illustrated in our experiments in Figure 6.2.

The Sparse Feature Learning Conjecture for More General Models

Results for diagonal linear nets recalled in the previous paragraph show that the noisy dynamics (6.9) induce a *sparsity bias*. As emphasized in HaoChen et al. (2020), this effect is largely due to the multiplicative structure of the noise $v \odot [X^\top \mathrm{d}B_t]$ that, in this case, has a shrinking effect *on the coordinates* (because of the coordinate-wise multiplication with v). In the general case, we see, thanks to (6.8), that the same multiplicative structure of the noise still happens but this time *with respect to the Jacobian* $\phi_\theta(X)$. Hence, this suggests that similarly to the diagonal linear network case, the implicit bias of the noise can lead to a shrinkage effect applied to $\phi_\theta(X)$. This effect depends on the noise intensity δ and the step size of SGD. Indeed, an interesting property of Brownian motion is that, for $v \in \mathbb{R}^p$, $\langle v, B_t \rangle = \|v\|_2 W_t$, where the equality holds in law and $(W_t)_{t\geq 0}$ is a one-dimensional Brownian motion. Hence, the process (6.8) is equivalent to a process whose i-th coordinate is driven by a noise proportional to $\|\phi_i\| \mathrm{d}W_t^i$, where ϕ_i is the i-th column of $\phi_\theta(X)$ and $(W_t^i)_{t\geq 0}$ is a Brownian motion. This SDE structure, similar to the geometric Brownian motion, is expected to induce the shrinkage of each multiplicative factor (Oksendal, 2013, Section 5.1), i.e., in our case $(\|\nabla_\theta h(x_i)\|)_{i=1}^n$. Thus, we conjecture:

The noise part of (6.8) *seeks to minimize the* ℓ_2*-norm of the columns of* $\phi_\theta(X)$.

Note that the *fitting part* of the dynamics prevents the Jacobian to collapse totally to zero, but as soon as they are not needed to fit the signal, *columns* can be reduced to zero. Remarkably, from a stability perspective, Blanc et al. (2020) showed a similar bias: locally around a minimum, the SGD dynamics implicitly tries to minimize the *Frobenius norm* $\|\phi_\theta(X)\|_{\mathrm{F}} = \sum_{i=1}^n \|\nabla_\theta h_\theta(x_i)\|^2$. Resolving the above conjecture and characterizing the implicit bias *along* the trajectory of SGD remains an exciting avenue for future work. Now, we provide a specification of this implicit bias for different architectures:

- **Diagonal linear networks:** For $h_{u,v}(x) = \langle u \odot v, x \rangle$, we have $\nabla_{u,v} h_{u,v}(x) = [v \odot x, u \odot x]$. Thus, for a generic data matrix X, minimizing the norm of each column of $\phi_{u,v}(X)$ amounts to put the maximal number of zero coordinates and hence to minimize $\|u \odot v\|_0$.
- **ReLU networks:** We take the prototypical one hidden layer to exhibit the sparsification effect. Let $h_{a,W}(x) = \langle a, \sigma(Wx) \rangle$, then $\nabla_a h_{a,W}(x) = \sigma(Wx)$ and $\nabla_{w_j} h_{a,W}(x) =$

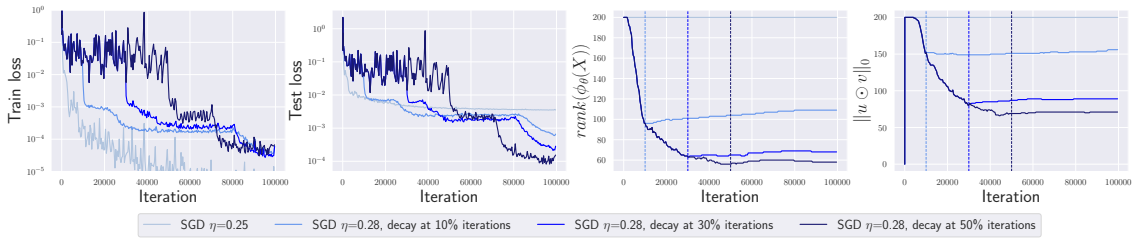


Figure 6.2: Diagonal linear networks. We observe loss stabilization, better generalization for longer schedules, minimization of the rank of $\phi_\theta(X)$ and sparsity of the predictor $u \odot v$.

$a_j x \mathbf{1}_{\langle w_j, x \rangle > 0}$. Note that the ℓ_2-norm of the column corresponding to the neuron is reduced when it is activated at a *minimal number of training points*, hence the implicit bias enables the learning of *sparse data-active features*. Finally, when some directions are needed to fit the data, similarly activated neurons align to fit, reducing the rank of $\phi_\theta(X)$.

Feature sparsity. Our main insight is that the Jacobian could be significantly simplified during the loss stabilization phase. Indeed, while the gradient part tries to fit the data and align neurons (see e.g. Fig. 6.10), the noise part of (6.8) intends to minimize the ℓ_2-norm of the columns of $\phi(X)$. Hence, in combination, this motivates us to count the average number of *distinct* (i.e., counting a group of aligned neurons as one), *non-zero* activations over the training set. We refer to this as the *feature sparsity coefficient* (see the next section for a detailed description). Note that the aforementioned sparsity comes both in the number of distinct neurons and their activation.

We show next that the conjectured sparsity is indeed observed empirically for a variety of models. Remark that both the feature sparsity coefficient and the rank of $\phi_\theta(X)$ can be used as a good proxy to track the hidden progress during the loss stabilization phase.

6.4 Empirical Evidence of Sparse Feature Learning Driven by SGD

Here we present empirical results for neural networks of increasing complexity: from diagonal linear networks to deep DenseNets on CIFAR-10, CIFAR-100, and Tiny ImageNet. We make the following common observations for all these networks trained using SGD schedules with large step sizes:

(**O1**) **Loss stabilization**: training loss stabilizes around a high level set until step size is decayed,

(**O2**) **Generalization benefit**: longer loss stabilization leads to better generalization,

(**O3**) **Sparse feature learning**: longer loss stabilization leads to sparser features.

Importantly, *we use no explicit regularization* (in particular, no weight decay) in our experiments so that the training dynamics is driven purely by SGD and the step size

schedule. Additionally, in some cases, we cannot find a single large step size that would lead to loss stabilization. In such cases, whenever explicitly mentioned, we use a *warmup* step size schedule—i.e., increasing step sizes according to some schedule—to make sure that the loss stabilizes around some level set. Warmup is commonly used in practice (He et al., 2016a; Devlin et al., 2018) and often motivated purely from the optimization perspective as a way to accelerate training (Agarwal et al., 2021), but we suggest that it is also a way to amplify the regularization effect of the SGD noise which is proportional to the step size.

Measuring sparse feature learning. We track the simplification of the Jacobian by measuring both the feature sparsity and the rank of $\phi_\theta(X)$. We compute the rank over iterations for each model (except deep networks for which it is prohibitively expensive) by using a fixed threshold on the singular values of $\phi_\theta(X)$ normalized by the largest singular value. In this way, we ensure that the difference in the rank that we detect is not simply due to different scales of $\phi_\theta(X)$. Moreover, we always compute $\phi_\theta(X)$ on the number of fresh samples equal to the number of parameters $|\theta|$ to make sure that rank deficiency is not coming from $n \ll |\theta|$ which is the case in the overparametrized settings we consider. To compute the **feature sparsity coefficient**, we count the average fraction of *distinct* (i.e., counting a group of highly correlated activations as one), *non-zero* activations at some layer over the training set. Note that the value of 100% means a completely *dense* feature vector and 0% means a feature vector with all zeros. We count a pair of activations i and j as highly correlated if their Pearson's correlation coefficient is at least 0.95. Unlike $\mathrm{rank}(\phi_\theta(X))$, the feature sparsity coefficient scales to deep networks and has an easy-to-grasp meaning.

6.4.1 Sparse Feature Learning in Diagonal Linear Networks

Setup. The inputs $x_1, \ldots, x_n$ with $n = 80$ are sampled from $\mathcal{N}(0, \boldsymbol{I}_d)$ where $\boldsymbol{I}_d$ is an identity matrix with $d = 200$, and the outputs are generated as $y_i = \langle \beta_*, x_i \rangle$ where $\beta_* \in \mathbb{R}^d$ is $r = 20$ sparse. We consider four different SGD runs (started from $u_i = 0.1$, $v_i = 0$ for each i): one with a small step size and three other with initial large step size decayed after 10%, 30%, 50% iterations, respectively.

Observations. We show the results in Fig. 6.2 and note that **(O1)**–**(O3)** hold even in this simple model trained with vanilla SGD without any explicit regularization or layer normalization schemes. We observe that the training loss stabilizes around $10^{-1.5}$, the test loss improves for longer schedules, both $\mathrm{rank}(\phi_\theta(X))$ and $\|u \odot v\|_0$ decrease during the loss stabilization phase leading to a sparse final predictor. While the training loss has seemingly converged to $10^{-1.5}$, a hidden dynamics suggested by (6.9) occurs which slowly drifts the iterates to a sparse solution. This implicit sparsification explains the dependence of the final test loss on the time when the large step size is decayed, similarly to what has been observed for deep networks in Fig. 6.1. Interestingly, we also note that SGD with large step-size schedules encounters saddle points *after* we decay the step size (see the training loss curves in Fig. 6.2) which resembles the saddle-to-saddle regime described in

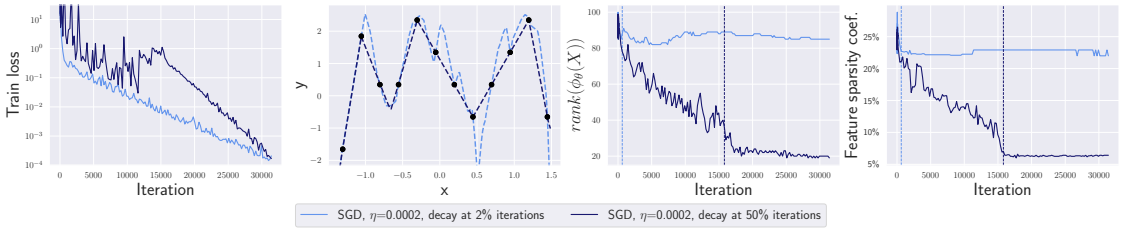


Figure 6.4: Two-layer ReLU networks for 1D regression. We observe loss stabilization, simplification of the model trained with a longer schedule, lower rank of $\phi_\theta(X)$, and much sparser features.

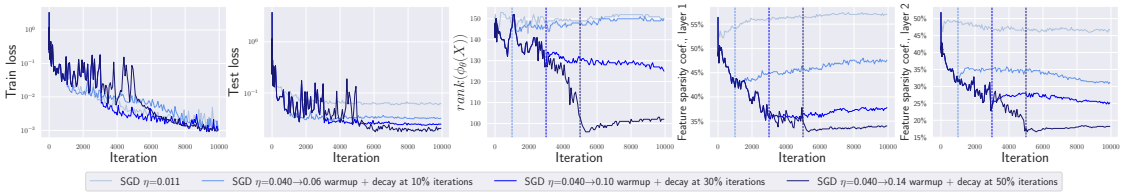


Figure 6.5: Three-layer ReLU networks in a teacher-student setup. We observe loss stabilization, lower rank of the Jacobian and lower feature sparsity coefficient on *both* hidden layers.

Jacot et al. (2021) which does not occur in the large-initialization lazy training regime.

SGD and GD have different implicit biases. Since we observe from Fig. 6.2 that for loss stabilization, stochasticity alone does not suffice and large step sizes are necessary, one may wonder if conversely, only large step sizes can be sufficient to have a sparsifying effect. Even if special instances can be found for which large step sizes are sufficient (such as for non-centered input features as in Nacson et al. (2022)), we answer this negatively showing that gradient descent in general does not go to the sparsest solution as demonstrated in Fig. 6.11 in the Appendix. Moreover, in Fig. 6.3, we visualize the difference in trajectory between the two methods taken with large step sizes over a 2D subspace spanned by $w^\star - w_{init}$ and $w_{flow} - w_{init}$, where $w^\star$ is the ground truth, w_{flow} is the result of gradient flow, and w_{init} is the initialization. This example provides an intuition that loss stabilization alone is not sufficient for sparsification and that the role of noise described earlier is crucial.

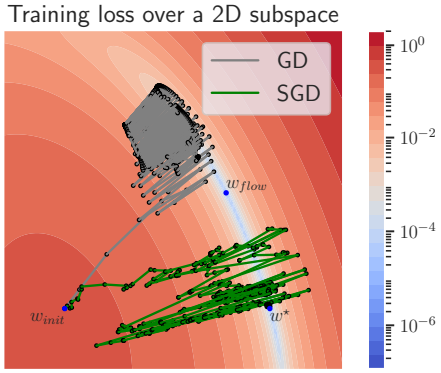


Figure 6.3: Diagonal linear networks. GD and SGD take different trajectories.

6.4.2 Sparse Feature Learning in Simple ReLU Networks

Two-layer ReLU network in 1D. We consider the 1D regression task from Blanc et al. (2020) with 12 points, where label noise SGD has been shown to learn a simple

model. We show that similar results can be achieved with large-step-size SGD via loss stabilization. We train a ReLU network with 100 neurons with SGD with a long linear warmup (otherwise, we were unable to achieve approximate loss stabilization), directly followed by a step size decay. The two plots correspond to a warmup/decay transition at 2% and 50% of iterations, respectively. The results shown in Fig. 6.4 confirm that **(O1)**–**(O3)** hold: the training loss stabilizes around $10^{-0.5}$, the predictor becomes much simpler and is expected to generalize better, and both $\text{rank}(\phi_\theta(X))$ and the feature sparsity coefficient substantially decrease during the loss stabilization phase. Interestingly, the rank reduction of $\phi_\theta(X)$ occurs because of zero *activations*, and not because of zero *weights*. For this one-dimensional task, we can directly observe the final predictor which is sparse in terms of the number of distinct ReLU kinks (i.e., having a few piecewise-linear segments) as captured by the feature sparsity coefficient and the rank of the Jacobian. Interestingly, we also observed *overregularization* for even larger step sizes when we cannot fit all the training points (see Fig. 6.12 in Appendix). This phenomenon clearly illustrates how the capacity control is induced by the optimization algorithm: *the function class over which we optimize depends on the step size schedule*. Additionally, Fig. 6.13 in App. shows the evolution of the predictor over iterations. The general picture is confirmed: first the model is simplified during the loss stabilization phase and only then fits the data.

Deeper ReLU networks. We use a teacher-student setup with a random *three-layer* teacher ReLU network having 2 neurons on each hidden layer. The student network is overparametrized with 10 neurons on each layer and is trained on 50 examples. Such teacher-student setup is useful since we know that the student network can implement the ground truth function but might not find it due to the small sample size. We train models using SGD with a medium constant step size and a large step size with warmup decayed after 10%, 30%, 50% iterations, respectively. The results shown in Fig. 6.5 confirm that **(O1)**–**(O3)** hold: the training loss stabilizes around $10^{-1.5}$, the test loss is smaller for longer schedules, and both $\text{rank}(\phi_\theta(X))$ and the feature sparsity coefficient substantially decrease during the loss stabilization phase. All methods have the same value of the training loss (10^{-3}) after 10^4 iterations but different generalization. Moreover, we see that the feature sparsity coefficient decreases *on each layer* which makes this metric a promising one to consider for deeper networks.

6.4.3 Sparse Feature Learning in Deep ReLU Networks

Setup. We consider here an image classification task and train a DenseNet-100-12 on CIFAR-10, CIFAR-100, and Tiny ImageNet using SGD with batch size 256 and different step size schedules. We use an exponentially increasing warmup schedule with exponent 1.05 to stabilize the training loss. We cannot measure the rank of $\phi(X)$ here since this matrix is too large ($\approx (5 \times 10^4) \times (2 \times 10^7)$) so we measure only the feature sparsity coefficient taken at two layers: at the end of super-block 3 (i.e., in the middle of the network) and super-block 4 (i.e., right before global average pooling at the end of the network) of DenseNets. We test two settings: a basic setting and a state-of-the-art setting

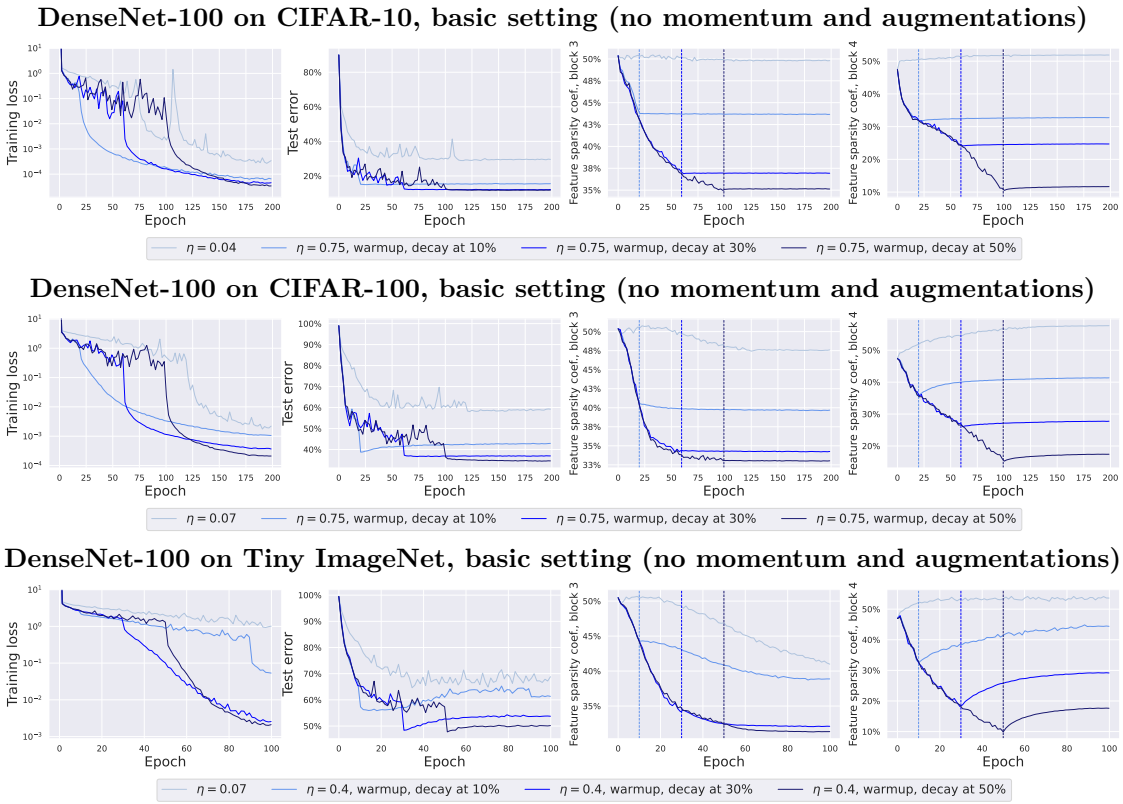


Figure 6.6: Experiments with DenseNet-100 in the basic setting. We can see that the training loss stabilizes, the test error noticeably depends on the length of the schedule, and the feature sparsity coefficient is minimized during the large step size phase.

with momentum and standard augmentations.

Observations. The results shown in Fig. 6.6 and 6.7 confirm that our main findings also hold for deep convolutional networks used in practice: the training loss approximately stabilizes, the test error is becoming progressively better for longer schedules, and the feature sparsity coefficient gradually decreases at both super blocks 3 and 4 until the step size is decayed. We also see that small step sizes consistently lead to suboptimal generalization, e.g., 60% vs. 35% in the basic setting on CIFAR-100. This poor performance confirms that it is crucial to leverage the implicit bias of large step sizes. The difference in the feature sparsity coefficient is also substantial: typically 50%-60% for small step sizes vs. 10%-20% for larger step sizes at block 4. The observations are similar for the state-of-the-art setting as well where we also see a noticeable difference in generalization and feature sparsity depending on the step size and schedule. Finally, we note that while both the feature sparsity coefficient and test error decrease together, it remains to be seen whether they are causally related on natural datasets.

We show the results with similar findings for other architectures (ResNets-18 and ResNets-34) on CIFAR-10 and CIFAR-100 in Fig. 6.15 and Fig. 6.16 in Appendix. Additionally, Fig. 6.17 illustrates that for small step sizes, the early and middle layers stay very close

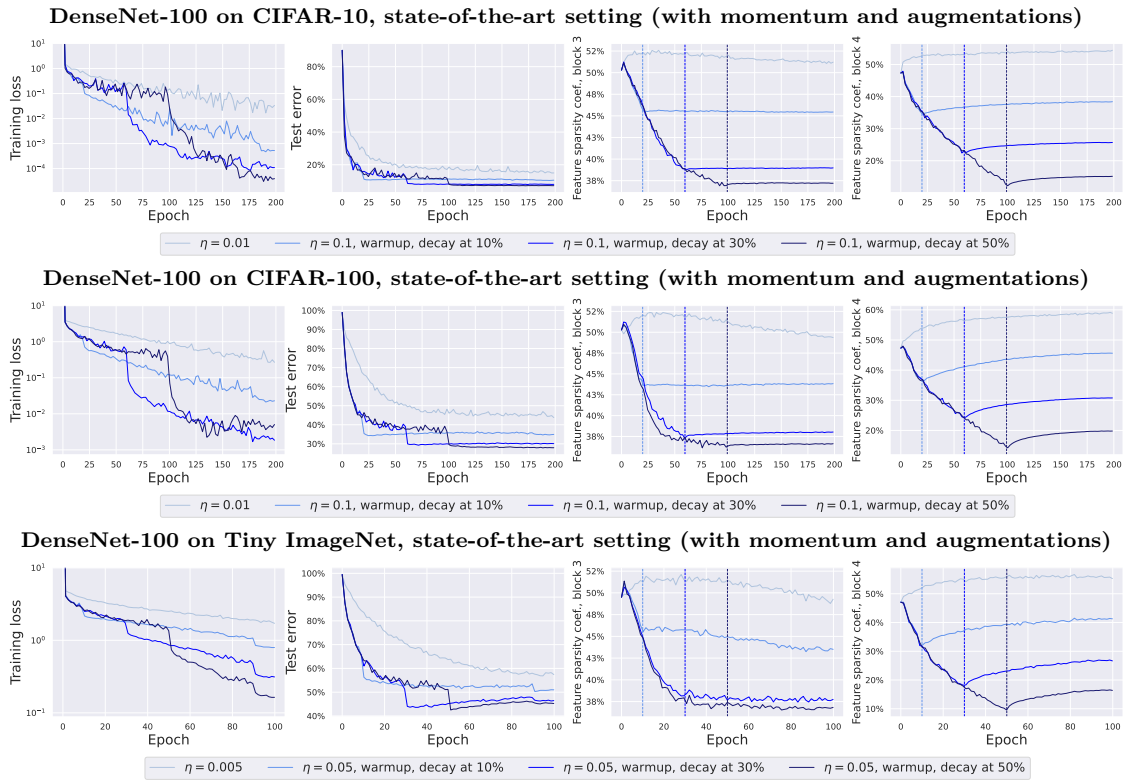


Figure 6.7: Experiments with DenseNet-100 in the state-of-the-art setting. We can see that the training loss stabilizes, the test error noticeably depends on the length of the schedule, and the feature sparsity coefficient is minimized during the large step size phase.

to their random initialization which indicates the absence of feature learning similarly to what is suggested by the neuron movement plot in Fig. 6.10 in the Appendix for a two-layer network in a teacher-student setup.

6.5 Conclusions and Insights from our Understanding of the Training Dynamics

Here we provide an extended discussion on the training dynamics of neural networks resulting from our theoretical and empirical findings.

The multiple stages of the SGD training dynamics. As analyzed and shown empirically, the training dynamics we considered can be split onto three distinct phases: (i) an initial phase of reducing the loss down to some level where stabilization can occur, (ii) a loss stabilization phase where noise and gradient directions combine to find architecture-dependent sparse representations of the data, (iii) a final phase when the step size is decreased to fit the training data. This typology clearly disentangles the effect of the stabilization phase (ii) which relies on the implicit bias of SGD to simplify the model. Note that phases (ii) and (iii) can be repeated until final convergence (He et al., 2016a). Moreover, in some training schedules, (ii) does not explicitly occur, and the effect of loss

stabilization (ii) and data fitting (iii) can occur simultaneously (Loshchilov and Hutter, 2019).

From lazy training to feature learning. Similar sparse implicit biases have been shown for regression with infinitely small initialization (Boursier et al., 2022) and for classification (Chizat and Bach, 2020a; Lyu and Li, 2020). However, both approaches are not practical from the computational point of view since (i) the origin is a saddle point for regression leading to the vanishing gradient problem (especially, for deep networks), and (ii) max-margin bias for classification is only expected to happen in the asymptotic phase (Moroshko et al., 2020). On the contrary, large step sizes enable to initialize far from the origin, while allowing to *efficiently* transition from a regime close to the lazy NTK regime (Jacot et al., 2018) to the rich feature learning regime.

Common patterns in the existing techniques. Tuning the step size to obtain loss stabilization can be difficult. To prevent early divergence caused by too large step sizes, we sometimes had to rely on an increasing step size schedule (known as *warmup*). Interpreting such schedules as a tool to favor implicit regularization provides a new explanation to their success and popularity. Additionally, normalization schemes like *batch normalization* or *weight decay*, beyond carrying their own implicit or explicit regularization properties, can be analyzed from a similar lens: they allow to use larger step sizes that boost further the implicit bias effect of SGD while preventing divergence (Bjorck et al., 2018; Zhang et al., 2018; Li and Arora, 2019). Note also that we derived our analysis with batch size equal to one for the sake of clarity, but an arbitrary batch size B would simply be equivalent to replacing $\gamma \leftarrow \gamma/B$. Similarly to the consequence of large step sizes, preferring *smaller batch sizes* (Keskar et al., 2016) while avoiding divergence seem key to benefit from the implicit bias of SGD. Finally, the effect of large step sizes or small batches is often connected to measures of *flatness* of the loss surface via stability analysis (Wu et al., 2018) and some methods like the Hessian regularization (Damian et al., 2021) or SAM (Foret et al., 2021) explicitly optimize it. Such methods resemble the implicit bias of SGD with loss stabilization implied by the label noise equation ((6.8)) where matrix $\phi_\theta(X)$ is the key component of the Hessian. However, an important practical difference is that the regularization strength in these methods is explicit and decoupled from the step size schedule which may be harder to properly tune since it is simultaneously responsible for optimization *and* generalization.

Appendix

In Section 6.6, we show Proposition 6.3.1 on the equivalence between SGD and GD with added noise. In Section 6.7, we provide the proof that loss stabilization occurs as written in Proposition 6.3.2. In Section 6.8, we show experimentally that the proposed SDE model matches well the SDE dynamics. Finally, we present additional experiments in Section 6.9.

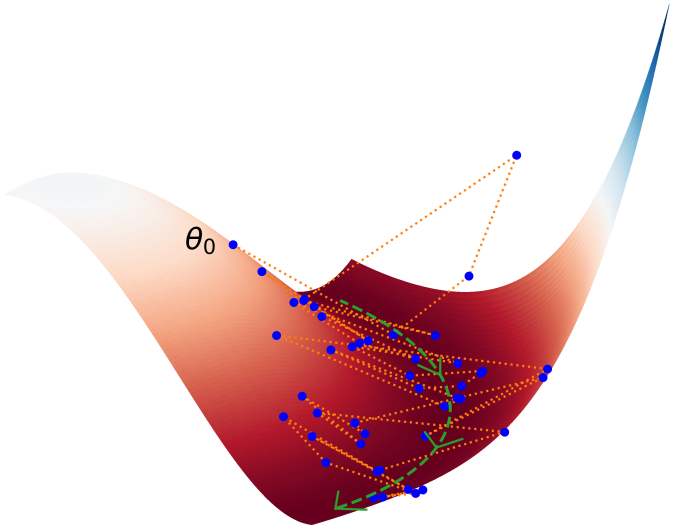


Figure 6.8: Three-dimensional visualisation of the SGD dynamics in a non-convex loss landscape. The SGD dynamics (blue points) is bouncing side-to-side to the bottom of the valley (the dotted green line). A slow movement occurs pushing the iterates in the direction given by the green arrows.

To begin this appendix, we provide in Figure 6.8 a toy visualization in which we showcase a typical SGD dynamics when loss stabilization occurs. We run SGD on the diagonal linear network with one sample in two dimensions ($n = 1, d = 2$) adding label noise of the shape given by equation (6.9), with balanced layers $u = v$. The blue points corresponds to iterates of the dynamics (that are linked with the orange dotted lines). The green line corresponds to the global minimum of the loss, what can be called the "bottom of the valley". This hopefully will serve the reader forge a visual intuition on (i) the bouncing dynamics side-to-side to the bottom of the valley (in green), and (ii) the slow stochastic movement (in the direction of the green arrows).

6.6 SGD and Label Noise GD

For the sake of clarity we recall below the statement of the Proposition 6.3.1 which we prove in this section.

Proposition 6.3.1. *Let $(\theta_t)_{t\geq 0}$ follow the SGD dynamics (6.2) with sampling function $(i_t)_{t\geq 0}$. Let $\mathbf{1}_{i=i_t}$ be indicator function, define for $t \geq 0$, the random vector $\xi_t \in \mathbb{R}^n$ such that for all $i \in [\![1, n]\!]$,*

$$[\xi_t]_i := (h_{\theta_t}(x_i) - y_i)(1 - n\mathbf{1}_{i=i_t}). \tag{6.10}$$

Then $(\theta_t)_{t\geq 0}$ follows the full-batch gradient dynamics on $\mathcal{L}$ with label noise $(\xi_t)_{t\geq 0}$, that is

$$\theta_{t+1} = \theta_t - \frac{\eta}{n}\sum_{i=1}^{n}(h_{\theta_t}(x_i) - y_i^t)\nabla_\theta h_{\theta_t}(x_i), \tag{6.11}$$

where we define the random labels $y^t := y + \xi_t$. Furthermore, ξ_t is a mean zero random vector with variance such that $\frac{1}{n(n-1)}\mathbb{E}\left\|\xi_t\right\|^2 = 2\mathcal{L}(\theta_t)$.

Proof. Note that

$$\sum_{i=1}^{n}(h_{\theta_t}(x_i) - y_i^t)\nabla_\theta h_{\theta_t}(x_i) = \sum_{i=1}^{n}(h_{\theta_t}(x_i) - y_i - [\xi_t]_i)\nabla_\theta h_{\theta_t}(x_i). \tag{6.12}$$

Using $[\xi_t]_i := (h_{\theta_t}(x_i) - y_i)(1 - n\mathbf{1}_{i=i_t})$,

$$= \frac{1}{n}\sum_{i=1}^{n}(h_{\theta_t}(x_i) - y_i - (h_{\theta_t}(x_i) - y_i)(1 - n\mathbf{1}_{i=i_t}))\nabla_\theta h_{\theta_t}(x_i), \tag{6.13}$$

$$= \sum_{i=1}^{n}\mathbf{1}_{i=i_t}(h_{\theta_t}(x_i) - y_i)\nabla_\theta h_{\theta_t}(x_i) = (h_{\theta_t}(x_{i_t}) - y_{i_t})\nabla_\theta h_{\theta_t}(x_{i_t}). \tag{6.14}$$

which is exactly the stochastic gradient wrt to sample (x_{i_t}, y_{i_t}).

Now we prove the latter part of the proposition regarding the scale of the noise. Recall that, for all $i \leqslant n$, we have $[\xi_t]_i = (h_{\theta_t}(x_i) - y_i)(1 - n\mathbf{1}_{i=i_t})$, where $i_t \sim \mathcal{U}([\![1, n]\!])$. Now taking the expectation,

$$\mathbb{E}[\xi_t]_i = \mathbb{E}\left[(h_{\theta_t}(x_i) - y_i)(1 - n\mathbf{1}_{i=i_t})\right] = (h_{\theta_t}(x_i) - y_i)(1 - n\mathbb{E}\left[\mathbf{1}_{i=i_t}\right]) = 0, \tag{6.15}$$

as $\mathbb{E}\left[\mathbf{1}_{i=i_t}\right] = 1/n$. Coming to the variance,

$$\mathbb{E}\left\|\xi_t\right\|^2 = \mathbb{E}\left[\sum_{i=1}^{n}[\xi_t]_i^2\right] = \sum_{i=1}^{n}\mathbb{E}\left[\xi_t\right]_i^2 \tag{6.16}$$

$$= \sum_{i=1}^{n}(h_{\theta_t}(x_i) - y_i)^2\mathbb{E}\left[(1 - n\mathbf{1}_{i=i_t})^2\right] \tag{6.17}$$

$$= \sum_{i=1}^{n}(h_{\theta_t}(x_i) - y_i)^2\mathbb{E}\left[(1 - 2n\mathbf{1}_{i=i_t} + n^2\mathbf{1}_{i=i_t})\right] \tag{6.18}$$

$$= \sum_{i=1}^{n}(h_{\theta_t}(x_i) - y_i)^2(1 - 2 + n) \tag{6.19}$$

$$= (n-1)\sum_{i=1}^{n}(h_{\theta_t}(x_i) - y_i)^2 = 2n(n-1)\mathcal{L}(\theta_t), \tag{6.20}$$

and this concludes the proof of the proposition. □

6.7 Quadratic Parameterization in One Dimension

Again, for the Appendix to be self-contained, we recall the setup of the Proposition 6.3.2 on loss stabilization. We consider a regression problem with quadratic parameterization on one-dimensional data inputs x_i's, coming from a distribution $\hat{\rho}$, and outputs generated by the linear model $y_i = x_i\theta_*^2$. The loss writes $F(\theta) := \frac{1}{4}\mathbb{E}_{\hat{\rho}}\left(y - x\theta^2\right)^2$, and the SGD iterates with step size $\eta > 0$ follow, for any $t \in \mathbb{N}$,

$$\theta_{t+1} = \theta_t + \eta\,\theta_t\,x_{i_t}\left(y_{i_t} - x_{i_t}\theta_t^2\right) \qquad \text{where} \qquad x_{i_t} \sim \hat{\rho}. \tag{6.21}$$

We rewrite the proposition here.

Proposition 6.7.1. (Extended version of Proposition 6.3.2) Assume $\exists\, x_{\min}, x_{\max} > 0$ such that $\mathrm{supp}(\hat{\rho}) \subset [x_{\min}, x_{\max}]$. Then for any $\eta \in ((\theta_* x_{\min})^{-2}, 1.25(\theta_* x_{\max})^{-2})$, any initialization in $\theta_0 \in (0, \theta_*)$, for $t \in \mathbb{N}$, we have almost surely

$$F(\theta_t) \in \left(\epsilon_{\mathrm{o}}^2\;\theta_*^2, 0.17\;\theta_*^2\right). \tag{6.22}$$

where $\epsilon_{\mathrm{o}} = \min\left\{(\eta(\theta_* x_{\min})^2 - 1)/3, 0.02\right\}$. Also, almost surely, there exists $t, k > 0$ such that $\theta_{t+2k} \in (0.65\;\theta_*, (1-\epsilon_{\mathrm{o}})\;\theta_*)$ and $\theta_{t+2k+1} \in ((1+\epsilon_{\mathrm{o}})\;\theta_*, 1.162\;\theta_*)$.

Proof. Consider SGD recursion (6.21) and note that $y = x\theta_*^2$.

$$\theta_{t+1} = \theta_t + \eta\,\theta_t\,x(x\theta_*^2 - x\theta_t^2) \tag{6.23}$$

$$\theta_{t+1} = \theta_t + \eta\,\theta_t\,x^2\,(\theta_*^2 - \theta_t^2) \tag{6.24}$$

For the clarity of exposition, we consider the rescaled recursion of the original SGD recursion.

$$\theta_{t+1}/\theta_* = \theta_t/\theta_* + \eta\;\theta_*^2\;x^2\;\theta_t/\theta_*\left(1 - (\theta_t/\theta_*)^2\right), \tag{6.25}$$

and, by making the benign change $\theta_t \leftarrow \theta_t/\theta_*$, we focus on the stochastic recursion instead,

$$\theta_{t+1} = \theta_t + \gamma\theta_t(1 - \theta_t^2), \tag{6.26}$$

where $\gamma \sim \hat{\rho}_\gamma$ the pushforward of $\hat{\rho}$ under the application $z \to \eta\,\theta_*^2\,z^2$. Let $\Gamma := \mathrm{supp}(\hat{\rho}_\gamma)$, the support of the distribution of γ. From the range of η, it can be verified that $\Gamma \subseteq (1, 1.25)$. Now the proof of the theorem follows from Lemma 10. □

Lemma 9 (Bounded Region)**.** *Consider the recursion* (6.26)*, for* $\Gamma \subseteq (1, 1.25)$ *and* $0 < \theta_0 < 1$*, then for all* $t > 0$*,* $\theta_t \in (0, 1.162)$*.*

Proof. Consider a single step of (6.26), for some $\gamma \in (1, 1.25)$,

$$\theta_+ = \theta + \gamma\theta(1 - \theta^2)$$

The aim is to show that θ_+ stays in the interval $(0, 1.162)$. In order to show this, we do a casewise analysis.

For $\theta \in (0, 1]$: Since $0 < \theta \leq 1$, we have $\theta_+ \geq \theta > 0$. To prove the bound above, consider the following quantity,

$$\theta_{max} = \max_{\gamma \in (1,1.25)} \max_{\theta \in (0,1]} \theta + \gamma\theta(1 - \theta^2) \tag{6.27}$$

Say $h_\gamma(\theta) = \theta + \gamma\theta(1-\theta^2)$, note that $h'_\gamma(\theta) = 1+\gamma-3\gamma\theta^2$ and $h''_\gamma(\theta) = -6\gamma\theta < 0$. Hence, for any γ in our domain, the maximum is attained at $\theta_\gamma = \frac{1}{\sqrt{3}}\sqrt{\frac{1}{\gamma} + 1}$ and $h_\gamma(\theta_\gamma) = \frac{2(1+\gamma)^{3/2}}{3\sqrt{3\gamma}}$.

$$\max_{\gamma \in (1,1.25)} \max_{\theta \in (0,1]} \theta + \gamma\theta(1 - \theta^2) = \max_{\gamma \in (.5,1.25)} \frac{2(1+\gamma)^{3/2}}{3\sqrt{3\gamma}} \tag{6.28}$$

It can be verified that $\frac{2(1+\gamma)^{3/2}}{3\sqrt{3\gamma}}$ is increasing with gamma in the interval $(1, 1.25)$. Hence,

$$\max_{\gamma \in (1,1.25)} \frac{2(1+\gamma)^{3/2}}{3\sqrt{3\gamma}} \leq \left.\frac{2(1+\gamma)^{3/2}}{3\sqrt{3\gamma}}\right|_{\gamma=1.25} < 1.162 \tag{6.29}$$

Combining them, we get,

$$\theta_+ \leq \max_{\gamma \in (0,1.25)} \max_{\theta \in (0,1]} \theta + \gamma\theta(1 - \theta^2) < 1.162 \tag{6.30}$$

For $\theta \in (1, 1.162)$: Since $\theta > 1$, we have, $\theta_+ < \theta < 1.162$. For lower bound, note that for θ_+ to be less than 0, we need $1 + \gamma - \gamma\theta^2 < 0$. But for $\gamma \in (1, 1.25)$ and $\theta \in (1, 1.162)$,

$$\gamma(\theta^2 - 1) < 1.25((1.162)^2 - 1) < 1. \tag{6.31}$$

Hence, it never goes below 0. □

Lemma 10. *Consider the recursion (6.26) with $\Gamma \subseteq (1, 1.25)$ and θ_0 initialized uniformly in $(0, 1)$. Then, there exists $\epsilon_0 > 0$, such that for all $\epsilon < \epsilon_0$ there exists $t > 0$ such that for any $k > 0$,*

$$\theta_{t+2k} \in (0.65, 1 - \epsilon) \quad \textit{and} \quad \theta_{t+2k+1} \in (1 + \epsilon, 1.162) \tag{6.32}$$

almost surely.

Proof. Define $\gamma_{\min} > 1$ as the infimum of the support Γ. Let $\epsilon_o = \min\{^{(\gamma_{min}-1)}/_3, 0.02\}$. Note that $\epsilon_0 > 0$ as $\gamma_{\min} > 1$. Now for any $0 < \epsilon < \epsilon_o$, we have $\gamma_{\min}(2-\epsilon)(1-\epsilon) > 2$.

Divide the interval (0,1.162) into 4 regions, $\mathrm{I}_0 = (0, 0.65]$, $\mathrm{I}_1 = (0.65, 1-\epsilon)$, $\mathrm{I}_2 = [1-\epsilon, 1)$, $\mathrm{I}_3 = (1, 1.162)$. The strategy of the proof is that the iterates will eventually end up in I_1 and that once it ends up in I_1, it comes back to I_1 in 2 steps.

Let θ_0 be initialized uniformly random in $(0, 1)$. Consider the sequence $(\theta_t)_{t\geq 0}$ generated by

$$\theta_{t+1} = \mathsf{h}_{\gamma_t}(\theta_t) := \theta_t + \gamma_t \theta_t (1-\theta_t^2) \quad \text{where} \quad \gamma_t \sim \hat{\rho}_\gamma. \tag{6.33}$$

We prove the following facts **(P1)-(P4)**:

(P1) There exists $t \geq 0$ such that the $\theta_t \in \mathrm{I}_1 \cup \mathrm{I}_2 \cup \mathrm{I}_3$.

(P2) Let $\theta_t \in \mathrm{I}_3$, then $\theta_{t+1} \in \mathrm{I}_1 \cup \mathrm{I}_2$.

(P3) Let $\theta_t \in \mathrm{I}_2$, there exists $k > 0$ such that for $k' < k$, $\theta_{t+2k'} \in \mathrm{I}_2$ and $\theta_{t+2k} \in \mathrm{I}_1$.

(P4) When $\theta_t \in \mathrm{I}_1$, then for all $k \geq 0$, $\theta_{t+2k} \in \mathrm{I}_1$ and $\theta_{t+2k+1} \in (1+\epsilon, 1.162)$.

Proof of (P1)-(P4): Let $t \in \mathbb{N}$, note first that the event $\{\theta_t = 1\} = \cup_{k \leqslant t}\{\theta_k = 1 | \theta_{k-1} \neq 1\}$ and hence a finite union of zero measure sets. Hence $\{\theta_t = 1\}$ is a zero measure set and therefore we do not consider it below. For any other sequence, from the above four properties, we can conclude that the lemma holds.

Proof of P1: Assume that until time $t > 0$, the iterates are all in I_0, then we have

$$\theta_t = \theta_{t-1}(1 + \gamma(1-\theta_{t-1}^2)) \geq \theta_{t-1}(2 - \theta_{t-1}^2) > 1.5\ \theta_{t-1} > 1.5^t\ \theta_0 \tag{6.34}$$

Hence, the sequence eventually exits I_0. We know that it will stay bounded from Lemma 9, hence it will end up in $\mathrm{I}_1 \cup \mathrm{I}_2 \cup \mathrm{I}_3$.

Proof of P2: For any $\theta_t \in (1, 1.162)$, $1 < \gamma < 1.25$, since $\mathsf{h}_\gamma(.)$ is decreasing in (1,1.162), we have $\mathsf{h}_\gamma(1.162) < \mathsf{h}_\gamma(\theta_t) < \mathsf{h}_\gamma(1)$. Also $\mathsf{h}_\gamma(\theta)$ is linear in gamma with negative coefficient for $\theta > 1$. Hence it decreases as γ increases. Using this,

$$.652 = \mathsf{h}_{1.25}(1.162) < \mathsf{h}_\gamma(1.162) < \mathsf{h}_\gamma(\theta_t) < \mathsf{h}_\gamma(1) = 1. \tag{6.35}$$

Hence, $\theta_{t+1} \in \mathrm{I}_1 \cup \mathrm{I}_2$.

Proof of P3: The proof of this follows from Lemma 12.

Proof of P4: The proof of this follows from Lemma 14. □

Lemma 11. *For any $\theta \in \mathrm{I}_1 \cup \mathrm{I}_2$ and any $a, b \in \Gamma$, $\mathsf{h}_a(\mathsf{h}_b(\theta)) \in \mathrm{I}_1 \cup \mathrm{I}_2$,*

$$\mathsf{h}_{\gamma_{\max}}(\mathsf{h}_{\gamma_{\max}}(\theta)) \leq \mathsf{h}_a(\mathsf{h}_b(\theta)) \leq \mathsf{h}_{\gamma_{\min}}(\mathsf{h}_{\gamma_{\min}}(\theta)). \tag{6.36}$$

Proof. For any $\gamma \in \Gamma$, recall

$$\mathsf{h}_\gamma(\theta) = \theta + \gamma\theta(1-\theta^2) = 1 + (1-\theta)(\gamma\theta(1+\theta) - 1). \tag{6.37}$$

Note that for $\theta \in \mathrm{I}_1 \cup \mathrm{I}_2$, $\theta(1+\theta) > 1$, Hence $\gamma\theta(1+\theta) > 1$. This gives us that $\mathsf{h}_\gamma(\theta) > 1$. Now we will track where $\theta \in \mathrm{I}_1 \cup \mathrm{I}_2$ can end up after two stochastic gradient steps.

- For any $b \in \Gamma$, as $\theta \in \mathrm{I}_1 \cup \mathrm{I}_2$, we have

  $$\mathsf{h}_{\gamma_{\max}}(\theta) \geq \mathsf{h}_b(\theta) \geq \mathsf{h}_{\gamma_{\min}}(\theta) > 1,$$

  note $\mathsf{h}_{\gamma_{\max}}(\theta) \geq \mathsf{h}_b(\theta) \geq \mathsf{h}_{\gamma_{\min}}(\theta)$ holds since $\theta < 1$.

- Now for any $a \in \Gamma$ and $x > 1$, $\mathsf{h}_a(x)$ is a decreasing function in x. Hence

  $$\mathsf{h}_a(\mathsf{h}_{\gamma_{\max}}(\theta)) \leq \mathsf{h}_a(\mathsf{h}_b(\theta)) \leq \mathsf{h}_a(\mathsf{h}_{\gamma_{\min}}(\theta)).$$

  Using $\gamma_{\min} \leq a$, $\mathsf{h}_a(\mathsf{h}_{\gamma_{\min}}(\theta)) \leq \mathsf{h}_{\gamma_{\min}}(\mathsf{h}_{\gamma_{\min}}(\theta))$, Similarly using $\gamma_{\max} > a$, we have, $\mathsf{h}_{\gamma_{\max}}(\mathsf{h}_{\gamma_{\max}}(\theta)) \leq \mathsf{h}_a(\mathsf{h}_{\gamma_{\max}}(\theta))$. Combining them we get,

  $$\mathsf{h}_{\gamma_{\max}}(\mathsf{h}_{\gamma_{\max}}(\theta)) \leq \mathsf{h}_a(\mathsf{h}_b(\theta)) \leq \mathsf{h}_{\gamma_{\min}}(\mathsf{h}_{\gamma_{\min}}(\theta)). \tag{6.38}$$

Similar argument can extend it to,

$$\mathsf{h}_{1.25}(\mathsf{h}_{1.25}(\theta)) < \mathsf{h}_a(\mathsf{h}_b(\theta)) < \mathsf{h}_1(\mathsf{h}_1(\theta)). \tag{6.39}$$

□

Lemma 12. *Let $\theta_t \in \mathrm{I}_2$, there exists $k > 0$ such that $\theta_{t+2k} \in \mathrm{I}_1$.*

Proof. For any $\gamma \in \Gamma$, let $\theta_+ = \mathsf{h}_\gamma(\theta)$, then we have

$$\mathsf{h}_\gamma(\mathsf{h}_\gamma(\theta)) - \theta = \mathsf{h}_\gamma(\theta_+) - \theta = \gamma\theta(1-\theta^2) + \gamma\theta_+(1-\theta_+^2). \tag{6.40}$$

Furthermore,

$$\theta_+ = \theta + \gamma\theta(1-\theta^2) = \theta(1+\gamma(1-\theta^2)), \tag{6.41}$$
$$1 + \theta_+ = 1 + \theta + \gamma\theta(1-\theta^2) = (1+\theta)(1+\gamma\theta(1-\theta)), \tag{6.42}$$
$$1 - \theta_+ = 1 - \theta - \gamma\theta(1-\theta^2) = (1-\theta)(1-\gamma\theta(1+\theta)). \tag{6.43}$$

And multiplying the above three terms and adding $\theta(1-\theta^2)$, we get,

$$\theta_+(1-\theta_+^2) + \theta(1-\theta^2) = \theta(1-\theta^2)\{1 + \underbrace{\Big[(1+\gamma(1-\theta^2))(1+\gamma\theta(1-\theta))(1-\gamma\theta(1+\theta))\Big]}_{P(\theta)}\} \tag{6.44}$$

For $\theta \in \mathrm{I}_2$, using $\gamma_{\min}(2-\epsilon)(1-\epsilon) > 2$, we have the inequalities

$$(1+\gamma(1-\theta^2))(1+\gamma\theta(1-\theta)) > 1, \tag{6.45}$$
$$(1-\gamma\theta(1+\theta)) < 1-\gamma_{min}(2-\epsilon)(1-\epsilon) < -1, \tag{6.46}$$
$$P(\theta) < -1. \tag{6.47}$$

Hence,

$$\mathsf{h}_\gamma(\mathsf{h}_\gamma(\theta)) - \theta = \gamma(1-\theta^2)(1+P(\theta)) < 0. \tag{6.48}$$

Therefore, for $[1-\epsilon, 1)$, for any $\gamma \in \Gamma$, $\mathsf{h}_\gamma(\mathsf{h}_\gamma(\theta)) < \theta$. Hence for any two stochastic gradient step with $a, b \in \Gamma$, from (6.36), $\theta_{t+2} = \mathsf{h}_a(\mathsf{h}_b(\theta_t)) \leq \mathsf{h}_{\gamma_{\min}}(\mathsf{h}_{\gamma_{\min}}(\theta_t)) < \theta_t$. From any point in I_2, we have $|\theta_{t+2}-1| > |\theta_t - 1|$, for any $a, b \in \Gamma$. Intutively this means that in *two gradient steps* the iterates move *further away from 1* until it eventually leaves the interval I_2 as the sequence $\{\theta_{t+2k}\}_{k\geq 0}$ is strictly decreasing with no limit point in I_2. From Lemma 13 , we know that in two steps the iterates will never leave $\mathrm{I}_1 \cup \mathrm{I}_2$. Hence they will eventually end up in I_1 leaving I_2. □

Property 6.7.1. *Define* $\mathsf{g}_\gamma(\theta) := \mathsf{h}_\gamma(\mathsf{h}_\gamma(\theta))$ *for the sake of brevity. The followings properties hold for* $\theta \in \mathrm{I}_1 \cup \mathrm{I}_2$, $\gamma \in \Gamma$ *and* θ_γ *the root of* $\mathsf{h}'_\gamma(\theta)$*:*

Q1 $\mathsf{g}_\gamma(\theta) \geq \mathsf{g}_\gamma(\theta_\gamma)$.

Q2 *The function* $\mathsf{g}_\gamma(.)$ *is decreasing in* $[0.65, \theta_\gamma)$ *and increasing in* $(\theta_\gamma, 1]$.

Proof. Note $\mathsf{h}'_\gamma(\theta) = 1+\gamma-\gamma 3\theta^2$ has at most one root $\theta_\gamma \in (0,1)$. Note that for all $\gamma \in \Gamma$, $\theta_\gamma \in \mathrm{I}_1 \cup \mathrm{I}_2$. For any γ, $\mathsf{g}'_\gamma(\theta) = \mathsf{h}'_\gamma(\mathsf{h}_\gamma(\theta))\mathsf{h}'_\gamma(\theta)$. For any $\theta \in \mathrm{I}_1 \cup \mathrm{I}_2$, we have, $\mathsf{h}_\gamma(\theta) > 1 \implies \mathsf{h}'_\gamma(\mathsf{h}_\gamma(\theta)) < 0$. Therefore, $\mathsf{g}'_\gamma(\theta)$ has only one root in $\mathrm{I}_1 \cup \mathrm{I}_2$. Since $\theta_\gamma \in \mathrm{I}_1 \cup \mathrm{I}_2$, note $\mathsf{g}''_\gamma(\theta_\gamma) = \mathsf{h}'_\gamma(\mathsf{h}_\gamma(\theta_\gamma))\mathsf{h}''_\gamma(\theta_\gamma) > 0$. Therefore, $\mathsf{g}_\gamma(.)$ attains its minimum at θ_γ and this shows the desired properties. □

Lemma 13. *For any* $\theta \in \mathrm{I}_1 \cup \mathrm{I}_2$ *and any* $a, b \in \Gamma$, $\mathsf{h}_a(\mathsf{h}_b(\theta)) \in \mathrm{I}_1 \cup \mathrm{I}_2$.

Proof. **Lower Bound:** From (6.39), we know

$$\mathsf{h}_{1.25}(\mathsf{h}_{1.25}(\theta)) < \mathsf{h}_a(\mathsf{h}_b(\theta)) \tag{6.49}$$

We know that from property **Q1** that $\mathsf{g}_\gamma(\theta) \geq \mathsf{g}_\gamma(\theta_\gamma)$. Hence

$$\mathsf{g}_{1.25}(\theta_{1.25}) < \mathsf{g}_{1.25}(\theta) < \mathsf{h}_a(\mathsf{h}_b(\theta)) \tag{6.50}$$

It can be quickly checked that $.65 < \mathsf{g}_{1.25}(\theta_{1.25})$. Hence the lower bound holds.

Upper Bound: From (6.39), we know

$$\mathsf{h}_a(\mathsf{h}_b(\theta)) < \mathsf{h}_1(\mathsf{h}_1(\theta)) \tag{6.51}$$

We know that from property **Q2** that $\mathsf{g}_1(\theta) \leq \max\{\mathsf{g}_1(1), \mathsf{g}_1(0.65)\}$. It can be easily verified that $\mathsf{g}_1(0.65) < 0.98$. Hence $\mathsf{g}_1(\theta) < 1$.

□

Lemma 14. *For any $\theta \in \mathrm{I}_1$ and any $a, b \in \Gamma$, $\mathsf{h}_a(\mathsf{h}_b(\theta)) \in \mathrm{I}_1$ and $\mathsf{h}_a(\theta) \in (1+\epsilon, 1.162)$.*

Proof. The lower bound in Lemma 13 holds here. For the upper bound, from and (6.36),

$$\mathsf{h}_a(\mathsf{h}_b(\theta)) \leq \mathsf{h}_{\gamma_{\min}}(\mathsf{h}_{\gamma_{\min}}(\theta)). \tag{6.52}$$

Using property **Q2**,

$$\mathsf{h}_{\gamma_{\min}}(\mathsf{h}_{\gamma_{\min}}(\theta)) \leq \max\{\mathsf{g}_{\gamma_{\min}}(1-\epsilon), \mathsf{g}_{\gamma_{\min}}(0.65)\} \tag{6.53}$$

From (6.48), $\mathsf{g}_{\gamma_{\min}}(1-\epsilon) < 1-\epsilon$. From (6.39), $\mathsf{g}_{\gamma_{\min}}(0.65) < \mathsf{g}_1(0.65) < 0.98 < 1-\epsilon$. In I_1, the function $\mathsf{h}_a(.)$ first increases reaches maximum and decreases. Hence for $\theta \in \mathrm{I}_1$, $\mathsf{h}_a(\theta) \geq \min\{\mathsf{h}_a(0.65), \mathsf{h}_a(1-\epsilon)\}$.

$$\begin{aligned}
\mathsf{h}_a(1-\epsilon) &\geq 1-\epsilon + a(1-(1-\epsilon)^2)(1-\epsilon), && (6.54)\\
&= 1-\epsilon + a(2\epsilon-\epsilon^2)(1-\epsilon), && (6.55)\\
&\geq 1-\epsilon + \gamma_{\min}(2\epsilon-\epsilon^2)(1-\epsilon), && (6.56)\\
&= 1+\epsilon+\epsilon\left(\gamma_{\min}(2-\epsilon)(1-\epsilon)-2\right) > 1+\epsilon. && (6.57)
\end{aligned}$$

Also $\mathsf{h}_a(0.65) > \mathsf{h}_1(0.65) > 1.02 > 1+\epsilon$, therefore $\mathsf{h}_a(\theta) > 1+\epsilon$ and this completes the proof. □

6.8 Empirical Validation of the SDE Modeling

In this section, we experimentally check the validity of the SDE modeling of SGD in (6.8) in terms of the key metrics: training loss, test loss, rank of the Jacobian, and feature sparsity.

SDE discretization. Let γ_t be the SDE discretization step size, η_t the step size of the corresponding SGD that we aim to validate, δ_t the noise intensity level, and $Z_t \sim \mathcal{N}(0, I_n)$. Then we discretize the SDE from (6.8) as follows:

$$\theta_{t+1} = \theta_t - \gamma_t \nabla_\theta \mathcal{L}(\theta_t) + \sqrt{\gamma_t}\sqrt{\eta_t \delta_t}\, \phi_{\theta_t}(X)^\top Z_t. \tag{6.58}$$

To approximate continuous time, we use a small discretization step size $\gamma_t := \eta_t/10$ and run the discretization for $10\times$ longer than the corresponding SGD run. We use $\eta_t := \eta^{SGD}_{\lfloor t/10 \rfloor}$ and $\delta_t := c \cdot \mathcal{L}(\theta^{SGD}_{\lfloor t/10 \rfloor})$ where c is a constant that we select for each setting separately to match the training dynamics of the corresponding SGD run. In addition, we also evaluate

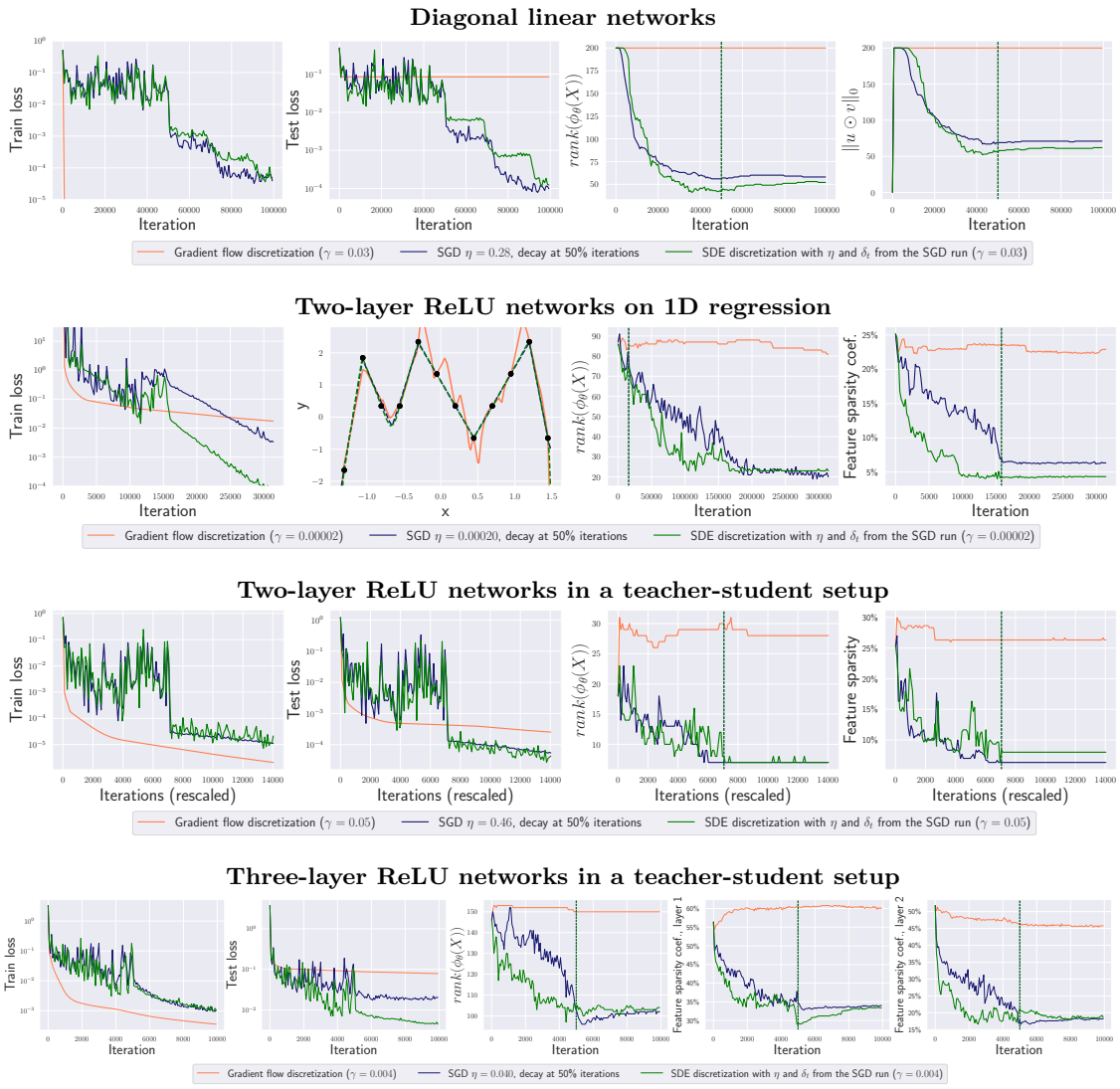


Figure 6.9: Empirical validation of the SDE modeling. In all cases, the dynamics of the SDE discretization qualitatively matches the dynamics of the corresponding SGD run. Moreover, gradient flow discretization exhibits no rank minimization or feature sparsity which suggests that the presence of the noise plays a key role in learning sparse features.

a discretization of gradient flow (i.e., (6.58) without the noise term) which helps to draw conclusions about the role of the noise term.

Experimental results. We present the discretization results in Fig. 6.9 for all models considered in the paper except deep networks for which computing the Jacobian ϕ_{θ_t} on each iteration of the SDE discretization is too costly. In all cases, the dynamics of the SDE discretization qualitatively matches the dynamics of the corresponding SGD run. In particular, we observe similar levels of decrease in the rank of the Jacobian and feature sparsity coefficient. We note that the match between SDE and SGD curves is not expected to be precise due to the inherent randomness of the process. Finally, we observe that gradient flow discretization exhibits no rank minimization or feature sparsity which suggests

that the presence of the noise (either from the original SGD or its SDE discretization) plays a key role in learning sparse features.

6.9 Additional Experimental Results

This section of the appendix presents additional experiments complementing the ones presented in the main text.

Illustration of neuron dynamics. We illustrate the change of neurons during training of two-layer ReLU networks (without biases) in the teacher-student setup of Chizat et al. (2019) (see Fig. 1 therein) using a large initialization scale for which small step sizes of GD or SGD lead to lazy training. We illustrate **(O1)**–**(O3)** in Fig. 6.14 and show neuron dynamics in Fig. 6.10. We see that for SGD with a small step size, the neurons w_i stay close to their initialization, while for a large step size, there is a clear clustering of directions w_i along the teacher directions $w_i^\star$. The overall picture is very similar to Fig. 1 of Chizat et al. (2019) where the same feature learning effect is achieved via gradient flow from a small initialization which is, however, much more computationally expensive due to the saddle point at zero. Finally, we note that the clustering phenomenon of neurons w_i motivates the removal of highly correlated activations in the feature sparsity coefficient: although the corresponding activations are often non-zero, many of them in fact implement *the same feature* and thus should be counted only once.

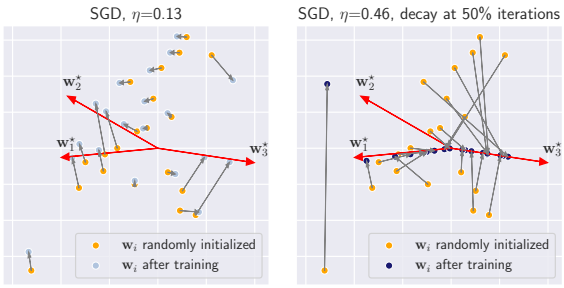


Figure 6.10: Only for a large step size, the neurons w_i cluster along the teacher neurons $w_i^\star$ leading to a model that uses a sparse set of features.

Further results. We give a short overview of additional figures referred to in the main text. More details can be found in the captions.

- Figure 6.11 shows that even if loss stabilization occurs in diagonal linear networks, the implicit bias towards sparsity is largely weaker than that of SGD and generalization is poor.
- Figures 6.12 and 6.13 demonstrate that the implicit bias resulting from high-loss stabilization makes the neural nets learn *first* a simple model *then eventually* fits the data.
- Figure 6.14 presents the sparsifying effect corresponding to the neurons' movements exhibited in Figure 6.10.
- Figures 6.15 and 6.16 exhibit the feature sparsity in ResNet-18 / ResNet-34 architectures on CIFAR-10 and CIFAR-100 in the basic and state-of-the-art settings.

- Figure 6.17 showcases the features learning induced by large step sizes for different layers of ResNets-18 when trained on CIFAR-10.

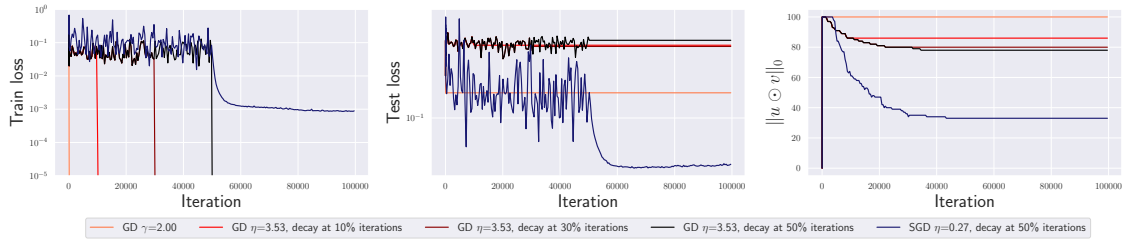


Figure 6.11: Diagonal linear networks. Loss stabilization also occurs for *full-batch gradient descent* but does not lead to a similar level of sparsity as SGD and also does not improve the test loss.

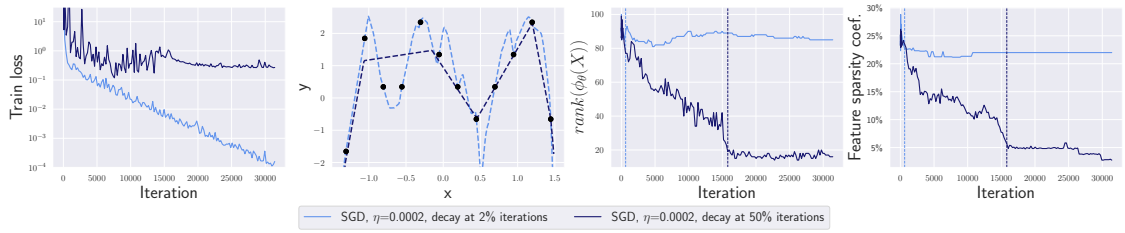


Figure 6.12: Two-layer ReLU networks for 1D regression. Unlike for Fig. 6.4, here we use a larger warmup coefficient (500× vs. 400×) which leads to overregularization such that the 50%-schedule run fails to fit all the training points and gets stuck at a too high value of the training loss ($\approx 10^{-0.5}$).

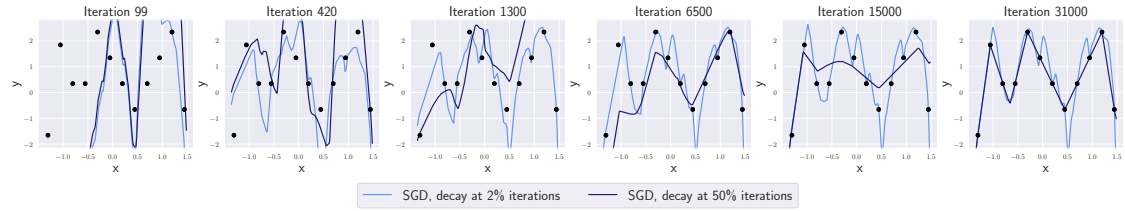


Figure 6.13: Two-layer ReLU networks for 1D regression. Illustration of the resulting models from Fig. 6.4 over training iterations. We can see that first the model is simplified and only then it fits the training data.

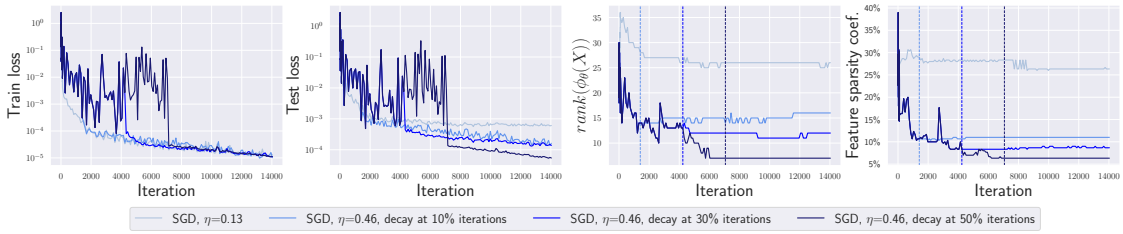


Figure 6.14: Two-layer ReLU networks in a teacher-student setup. Loss stabilization for two-layer ReLU nets in the teacher-student setup with input dimension $d = 2$. We observe loss stabilization, better test loss for longer schedules and sparser features due to simplification of $\phi(X)$.

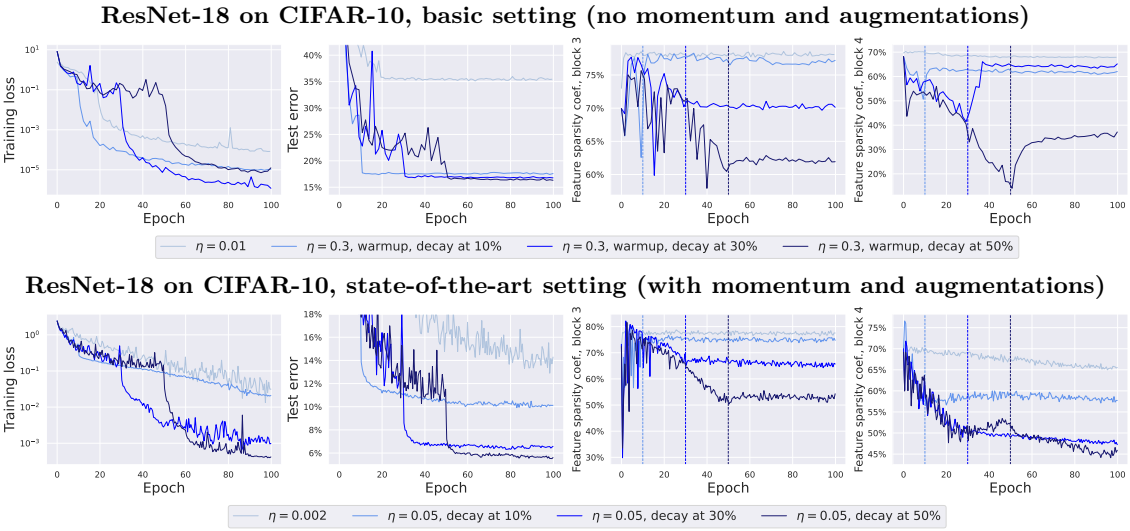


Figure 6.15: ResNet-18 trained on CIFAR-10. Both in the basic and state-of-the-art settings, the training loss stabilizes, the test loss noticeably depends on the length of the schedule, and the feature sparsity coefficient is minimized over iterations.

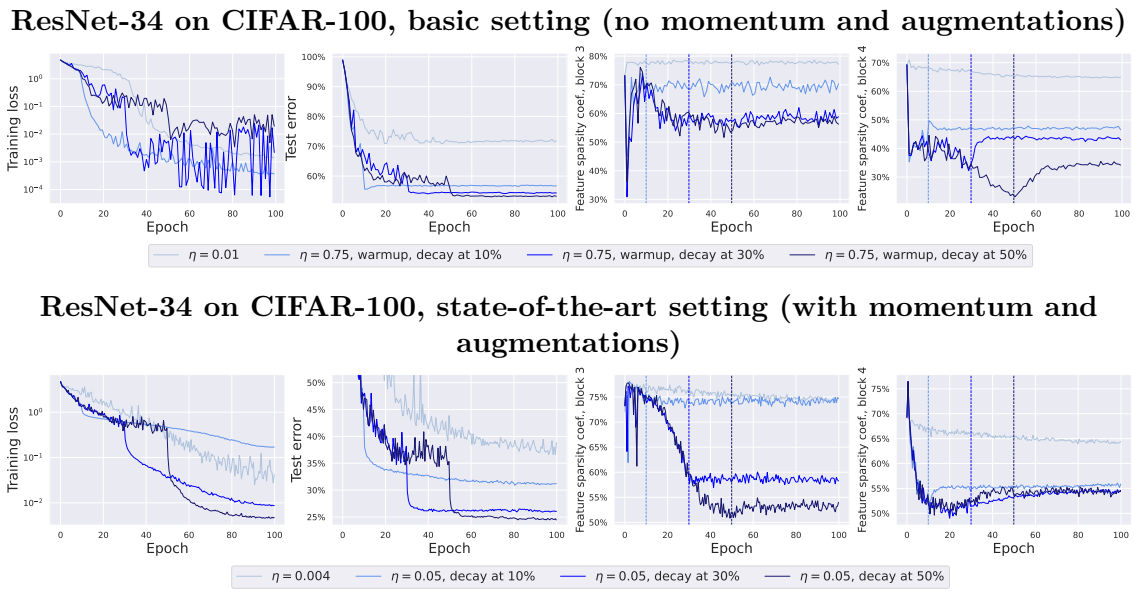


Figure 6.16: ResNet-34 trained on CIFAR-100. Both in the basic and state-of-the-art settings, the training loss stabilizes, the test loss significantly depends on the length of the schedule, and feature sparsity is minimized over iterations. However, differently from the plots on CIFAR-10, here without explicit regularization we observe oscillating behavior after the step size decay (although at a very low level between 10^{-4} and 10^{-2}).

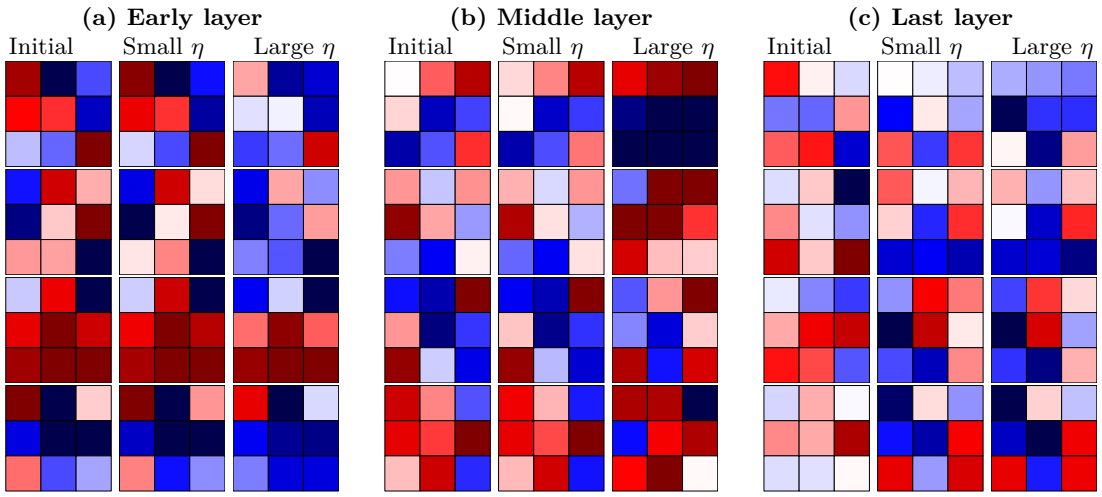


Figure 6.17: Visualization on four sets of convolutional filters taken from different layers of ResNets-18 trained on CIFAR-10 with small vs. large step size η (the 50% decay schedule). For small step sizes, the early and middle layers stay very close to randomly initialized ones which indicates the absence of feature learning.

7 A Modern Look at the Relationship between Sharpness and Generalization

7.1 Preface

In this chapter, based on Andriushchenko et al. (2023b), we provide a comprehensive study of how well different definitions of sharpness of minima correlate with generalization. We conclude that sharpness does not necessarily correlate well with generalization but rather with some training parameters like the learning rate.

Summary Sharpness of minima is a promising quantity that can positively correlate with test error for deep networks, and its minimization during training can improve generalization. However, standard sharpness is not invariant under reparametrizations of neural networks, and, to fix this, reparametrization-invariant sharpness definitions have been proposed, most prominently *adaptive sharpness* (Kwon et al., 2021). But does it really capture generalization in modern practical settings? We comprehensively explore this question in a detailed study of various definitions of adaptive sharpness in settings ranging from training from scratch on ImageNet and CIFAR-10 to fine-tuning CLIP on ImageNet and BERT on MNLI. We focus mostly on *transformers* for which little is known in terms of sharpness despite their widespread usage. Overall, we observe that sharpness *does not correlate well* with generalization but rather with some training parameters like the learning rate that can be positively or negatively correlated with generalization depending on the setup. Interestingly, in multiple cases, we observe a consistent *negative* correlation of sharpness with out-of-distribution error implying that *sharper* minima can generalize *better*. Finally, we illustrate on a simple model that the right sharpness measure is highly data-dependent, and that we do not understand well this aspect for realistic data distributions. Our code is available at https://github.com/tml-epfl/sharpness-vs-generalization.

Co-authors Francesco Croce, Maximilian Müller, Matthias Hein, Nicolas Flammarion.

Contributions Maksym Andriushchenko proposed the project, performed most of the experiments, and provided the theoretical analysis. Francesco Croce and Maximilian Müller contributed to the experiments.

7.2 Introduction

Considering the sharpness of the training objective at a minimum has intuitive appeal: if the loss surface is slightly perturbed due to a train vs. test or out-of-distribution (OOD) discrepancy, flat minima of deep networks should still have low loss (Hochreiter and Schmidhuber, 1995; Keskar et al., 2016). On the theoretical side, sharpness appears in generalization bounds (Neyshabur et al., 2017; Dziugaite and Roy, 2018; Foret et al., 2021) but this fact alone is not necessarily informative for practical settings. For example, quantities like the VC-dimension typically correlate *negatively* with generalization contrary to what the generalization bound might suggest (Jiang et al., 2019). Importantly, it has been shown empirically that sharpness can also correlate well with generalization in common deep learning setups (Keskar et al., 2016; Jiang et al., 2019) which makes it a promising generalization measure that can potentially distinguish well-generalizing solutions. Additionally, empirical success of training methods that minimize sharpness such as sharpness-aware minimization (SAM) (Zheng et al., 2021; Wu et al., 2020b; Foret et al., 2021) further suggests that sharpness can be an important quantity for generalization.

Motivation: why revisiting sharpness? Many works imply or conjecture that flatter minima should generalize better (Xing et al., 2018; Zhou et al., 2020; Cha et al., 2021; Park and Kim, 2022; Lyu et al., 2022) for standard or OOD data. However, standard sharpness definitions do not correlate well with generalization (Jiang et al., 2019; Kaur et al., 2022) which can be partially due to their lack of invariance under reparametrizations that leave the model unchanged (Dinh et al., 2017; Granziol, 2020; Zhang et al., 2021b). Adaptive sharpness appears to be more promising since it fixes the reparametrization issue and is shown to empirically correlate better with generalization (Kwon et al., 2021). However, the empirical evidence in Kwon et al. (2021) and other works that discuss sharpness (Keskar et al., 2016; Jiang et al., 2019; Dziugaite et al., 2020; Bisla et al., 2022) is restricted to small datasets like CIFAR-10 or SVHN. In addition, SAM appears to be particularly useful for new architectures like vision transformers (Chen et al., 2022) for which there has been no systematic studies of sharpness vs. generalization. Moreover, transfer learning is becoming the default option for vision and language tasks but not much is known about sharpness there. Finally, the relationship between sharpness and OOD generalization is also underexplored. These new developments motivate us to revisit the role of sharpness in these new settings.

Contributions. We aim to provide a comprehensive study focusing specifically on adaptive sharpness in order to answer the following fundamental question:

Can reparametrization-invariant sharpness capture generalization in modern practical settings?

Towards this goal, we make the following contributions:

- We provide extensive evaluations of multiple reparametrization-invariant sharpness

measures for (1) training from scratch on ImageNet and CIFAR-10 using transformers and ConvNets, and (2) fine-tuning CLIP and BERT transformers on ImageNet and MNLI.

- We observe that sharpness *does not correlate well* with generalization but rather with some training parameters like the learning rate which can be positively or negatively correlated with generalization depending on the setup.
- Interestingly, in multiple cases, we observe a consistent *negative* correlation of sharpness with OOD generalization implying that *sharper* minima can generalize *better*.
- Finally, we provide an analysis on a simple model where we know the measure responsible for generalization. Our analysis suggests that (1) different sharpness definitions can capture totally different trends, and (2) the right sharpness measure is highly *data-dependent*.

7.3 Related work

Here we discuss the most related papers to our work.

Systematic studies on sharpness vs. generalization. The seminal work of Keskar et al. (2016) shows that the performance degradation of large-batch SGD (LeCun et al., 2012) is correlated with sharpness of minima. Neyshabur et al. (2017) explore different generalization measure that may explain generalization for deep networks suggesting that sharpness can be a promising measure. Jiang et al. (2019) perform a systematic study that shows a strong correlation between sharpness and generalization on a large set of CIFAR-10/SVHN models trained with many different hyperparameters. Their experimental protocol is, however, criticized in Dziugaite et al. (2020) since it can obscure failures of generalization measures and instead should be evaluated within the framework of distributional robustness. Vedantam et al. (2021) discuss OOD generalization on small datasets and evaluate a definition of sharpness which, however, does not correlate well with OOD generalization. Stutz et al. (2021) study the relationship between sharpness and generalization under ℓ_p-bounded adversarial perturbations. Andriushchenko and Flammarion (2022) study reasons behind the success of SAM and highlight the importance of using sharpness computed on a small subset of training points. Kaur et al. (2022) discuss that the maximum eigenvalue of the Hessian is not always predictive to generalization even for models obtained via standard training methods.

Reparametrization-invariant sharpness definitions. The magnitude-aware sharpness of Keskar et al. (2016) mitigates but does not completely resolve reparametrization invariance. Liang et al. (2019) consider the Fisher-Rao metric related to sharpness and invariant to network reparametrization. Petzka et al. (2021) propose a sharpness measure based on the trace of the Hessian and show correlation for a small ConvNet on CIFAR-10. Tsuzuku et al. (2020) suggest to use a specifically rescaled sharpness inspired by the PAC-Bayes theory and report high correlation with generalization for ResNets on CIFAR-10.

Most importantly for our work, Kwon et al. (2021) introduce adaptive sharpness which is reparametrization invariant, correlates well with generalization, and generalizes multiple existing sharpness definitions.

Explicit and implicit sharpness minimization. The idea that flat minima can be beneficial for generalization dates back to Hochreiter and Schmidhuber (1995) and inspires multiple methods that optimize for more robust minima. These methods optimize different criteria ranging from random perturbations such as dropout (Srivastava et al., 2014) and Entropy-SGD (Chaudhari et al., 2016) to worst-case perturbations such as SAM (Foret et al., 2021) and its variations (Kwon et al., 2021; Zhuang et al., 2022; Du et al., 2022). Notably, Chen et al. (2022) suggest that SAM is particularly helpful for vision transformers on ImageNet scale and that standard transformers by default converge to very sharp minima. Concurrently, works on the implicit bias of SGD suggest *implicit* minimization of some hidden complexity measures related to flatness of minima (Keskar et al., 2016; Smith and Le, 2018; Xing et al., 2018). Izmailov et al. (2018) propose to average weights during SGD to improve generalization and motivate it by sharpness reduction. Smith et al. (2021) derive an implicit regularization term of SGD based on the gradient norm. Sharpness-related quantities based on the Hessian have been a focus of many recent works. E.g., Cohen et al. (2021); Arora et al. (2022); Damian et al. (2023) empirically and theoretically characterize the regime of full-batch gradient descent where the maximum eigenvalue of the Hessian becomes inversely proportional to the learning rate used for training. Blanc et al. (2020); Li et al. (2022); Damian et al. (2021) discover implicit minimization of the trace of the Hessian for label-noise SGD used as a proxy of standard SGD. The common theme behind these works is a focus on sharpness-related metrics as a tool to better understand generalization for deep networks.

7.4 Adaptive Sharpness, its Invariances, and Computation

In this section, we first provide background on adaptive sharpness, then discuss its invariance properties for modern architectures, and propose a way to compute worst-case sharpness efficiently.

7.4.1 Background on Sharpness

Sharpness definitions. We denote the loss on a set of *training* points $\mathcal{S}$ as $L_{\mathcal{S}}(\boldsymbol{w}) = \frac{1}{|S|}\sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{S}} \ell_{\boldsymbol{xy}}(\boldsymbol{w})$, where $\ell_{\boldsymbol{xy}}(\boldsymbol{w}) \in \mathbb{R}_+$ represents some loss function (e.g., cross-entropy) on the training pair $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{S}$ computed with the network weights $\boldsymbol{w}$. For arbitrary $\boldsymbol{w} \in \mathbb{R}^p$ (i.e., not necessarily a minimum), we define the *adaptive average-case* and *adaptive worst-case m-sharpness* with radius ρ and with respect to a vector $\boldsymbol{c} \in \mathbb{R}^p$ as:

$$
\begin{aligned}
S^{\rho}_{avg}(\boldsymbol{w}, \boldsymbol{c}) &\triangleq \mathbb{E}_{\substack{\mathcal{S}\sim P_m \\ \boldsymbol{\delta}\sim\mathcal{N}(0,\rho^2 diag(\boldsymbol{c}^2))}} L_{\mathcal{S}}(\boldsymbol{w}+\boldsymbol{\delta}) - L_{\mathcal{S}}(\boldsymbol{w}), \\
S^{\rho}_{max}(\boldsymbol{w}, \boldsymbol{c}) &\triangleq \mathbb{E}_{\mathcal{S}\sim P_m} \max_{\|\boldsymbol{\delta}\odot\boldsymbol{c}^{-1}\|_p \leq \rho} L_{\mathcal{S}}(\boldsymbol{w}+\boldsymbol{\delta}) - L_{\mathcal{S}}(\boldsymbol{w}),
\end{aligned}
\tag{7.1}
$$

where $\odot/^{-1}$ denotes elementwise multiplication/inversion and P_m is the data distribution that returns m training pairs $(\boldsymbol{x}, \boldsymbol{y})$. Both average-case and worst-case sharpness have often been considered in the literature, and worst-case sharpness is mostly determined to correlate better with generalization (Jiang et al., 2019; Dziugaite et al., 2020; Kwon et al., 2021), especially with a small m (i.e., $|\mathcal{S}|$) in worst-case sharpness (Foret et al., 2021). Using $\boldsymbol{c} = |\boldsymbol{w}|$ leads to *elementwise* adaptive sharpness (Kwon et al., 2021) and makes the sharpness invariant under multiplicative reparametrizations that preserve the network, i.e., for any $\boldsymbol{c} \in \mathbb{R}^p$ such that $f(\boldsymbol{w} \odot \boldsymbol{c}) = f(\boldsymbol{w})$ we have:

$$
\begin{aligned}
&S^{\rho}_{max}(\boldsymbol{w} \odot \boldsymbol{c}, |\boldsymbol{w} \odot \boldsymbol{c}|) = \\
&\mathbb{E}_{\mathcal{S}} \max_{\|\boldsymbol{\delta} \odot (|\boldsymbol{w}| \odot \boldsymbol{c})^{-1}\|_p \leq \rho} L_{\mathcal{S}}(\boldsymbol{w} \odot \boldsymbol{c} + \boldsymbol{\delta}) - L_{\mathcal{S}}(\boldsymbol{w} \odot \boldsymbol{c}) = \\
&\mathbb{E}_{\mathcal{S}} \max_{\|\boldsymbol{\delta}' \odot |\boldsymbol{w}|^{-1}\|_p \leq \rho} L_{\mathcal{S}}((\boldsymbol{w} + \boldsymbol{\delta}') \odot \boldsymbol{c}) - L_{\mathcal{S}}(\boldsymbol{w} \odot \boldsymbol{c}) = \\
&\mathbb{E}_{\mathcal{S}} \max_{\|\boldsymbol{\delta}' \odot |\boldsymbol{w}|^{-1}\|_p \leq \rho} L_{\mathcal{S}}(\boldsymbol{w} + \boldsymbol{\delta}') - L_{\mathcal{S}}(\boldsymbol{w}) = S^{\rho}_{max}(\boldsymbol{w}, |\boldsymbol{w}|),
\end{aligned}
$$

where we used the substitution $\boldsymbol{\delta}' := \boldsymbol{\delta} \odot \boldsymbol{c}^{-1}$. Similarly, one can show that $S^{\rho}_{avg}(\boldsymbol{w} \odot \boldsymbol{c}, |\boldsymbol{w} \odot \boldsymbol{c}|) = S^{\rho}_{avg}(\boldsymbol{w}, |\boldsymbol{w}|)$. Thus, this illustrates that *the criticism of sharpness stated in Dinh et al. (2017) does not apply to adaptive sharpness*, and there is no need to "balance" the network in a pre-processing step like, e.g., done in Bisla et al. (2022).

Connections between different sharpness definitions. Here we generalize the analytical expressions of standard sharpness for radius $\rho \to 0$ that depend on the first- or second-order terms which are frequently used in the literature (Blanc et al., 2020; Tsuzuku et al., 2020; Li et al., 2022; Damian et al., 2021). For a thrice differentiable loss $L(\boldsymbol{w})$, the average-case elementwise adaptive sharpness can be computed as (see App. 7.8.1 for proofs):

$$
S^{\rho}_{avg}(\boldsymbol{w}, |\boldsymbol{w}|) = \mathbb{E}_{\mathcal{S} \sim P_m} \frac{\rho^2}{2} \mathrm{tr}(\nabla^2 L_{\mathcal{S}}(\boldsymbol{w}) \odot |\boldsymbol{w}||\boldsymbol{w}|^{\top}) + O(\rho^3). \tag{7.2}
$$

We note that the first-order term cancels out completely and plays no role. This is not the case for worst-case adaptive sharpness where we get for $p = 2$ the following expression for every critical point that is not a local maximum:

$$
S^{\rho}_{max}(\boldsymbol{w}, |\boldsymbol{w}|) = \mathbb{E}_{\mathcal{S} \sim P_m} \frac{\rho^2}{2} \lambda_{\max}(\nabla^2 L_{\mathcal{S}}(\boldsymbol{w}) \odot |\boldsymbol{w}||\boldsymbol{w}|^{\top}) + O(\rho^3), \tag{7.3}
$$

otherwise the first-order term dominates and we get $\rho\, \mathbb{E}_{\mathcal{S} \sim P_m} \|\nabla L(\boldsymbol{w}) \odot |\boldsymbol{w}|\|_2$, which resembles the implicit gradient regularization of Smith et al. (2021). Thus, worst-case sharpness with a small radius captures different properties of the loss surface depending on whether $\boldsymbol{w}$ is close to a minimum or not. We make use of these quantities in the last section to discuss insights from simple models. For the experiments, however, we evaluate a range of ρ where the smallest ρ well-approximates the above quantities.

What do we expect sharpness to capture? We are looking for a sharpness measure that can be *predictive for generalization* meaning that it satisfies either of these two hypotheses:

- **Strong hypothesis**: sharpness is highly correlated with generalization suggesting a *possibility* of a causal relation.
- **Weak hypothesis**: models with the lowest sharpness generalize well suggesting that sharpness might be *sufficient but not necessary* for generalization.

To detect correlation, we follow the previous works by Jiang et al. (2019); Dziugaite et al. (2020); Kwon et al. (2021) and use the Kendall rank correlation coefficient:

$$\tau(\boldsymbol{t}, \boldsymbol{s}) = \frac{2}{M(M-1)} \sum_{i<j} \operatorname{sign}(t_i - t_j) \operatorname{sign}(s_i - s_j) \tag{7.4}$$

where $\boldsymbol{t}, \boldsymbol{s} \in \mathbb{R}^M$ are vectors of test error and sharpness values for M different models. We adopt a less demanding setting than in the previous works of Neyshabur et al. (2017); Jiang et al. (2019); Dziugaite et al. (2020), and only compare models *within the same loss surface* motivated by the geometric motivation behind sharpness. This restriction rules out comparing models with different architectures (including different width and depth) or measuring sharpness on a different set of points since both changes would change the loss surface. According to the same reason, we also do not consider the ability of sharpness to capture robustness to different amounts of noisy labels (unlike, e.g., Neyshabur et al. (2017)). We always evaluate sharpness on the *same* training points taken without any data augmentations. Moreover, we always compare models trained with exactly the same training sets but, at the same time, we allow the usage of algorithmic techniques such as data augmentation or mixup for training.

7.4.2 Which Invariances Do We Need Sharpness to Capture for Modern Architectures?

Throughout the paper, we focus on *elementwise* adaptive sharpness which, as we show, satisfies the main reparametrization invariances for ResNets and ViTs. Let us denote $f_{\boldsymbol{w}} : \mathbb{R}^d \to \mathbb{R}^K$ a network with parameters $\boldsymbol{w}$, which returns the logits $f_{\boldsymbol{w}}(\boldsymbol{x}) \in \mathbb{R}^K$ for an input $\boldsymbol{x} \in \mathbb{R}^d$. By a reparametrization invariance we mean a function $T : \mathbb{R}^p \to \mathbb{R}^p$ such that for every $\boldsymbol{w} \in \mathbb{R}^p$ and $\boldsymbol{x} \in \mathbb{R}^d$ it holds $f_{\boldsymbol{w}}(\boldsymbol{x}) = f_{T(\boldsymbol{w})}(\boldsymbol{x})$. We briefly discuss here that adaptive sharpness also stays invariant for *modern* architectures like ResNets and ViTs involving normalization layers and self-attention. Finally, we discuss how to treat the scale-sensitivity of classification losses.

Adaptive sharpness for ResNets. A typical block of a pre-activation ResNet between skip connections includes the following sequence of operations: `BN`→`ReLU`→`conv`→`BN`→`ReLU`→`conv` where `BN` denotes BatchNorm. So we need to make sure that the sharpness

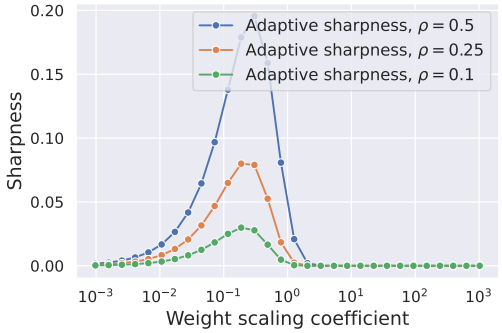


Figure 7.1: Sensitivity of adaptive sharpness to weight scaling for a linear model that achieves zero training error.

definition we use is invariant to transformations that leave the network unchanged: (1) multiplication of the affine BatchNorm parameters by $\alpha \in \mathbb{R}_+$ and division of the subsequent convolutional parameters by the same α (since ReLU is positive one-homogeneous and $\text{ReLU}(\alpha z)/\alpha = \text{ReLU}(z)$), and (2) multiplying the convolutional layer by any $\alpha \in \mathbb{R}_+$ due to scale-invariance of the subsequent BatchNorm layer. Both multiplicative invariances are satisfied by elementwise adaptive sharpness since $S^{\rho}_{max}(\boldsymbol{w} \odot \boldsymbol{c}, |\boldsymbol{w} \odot \boldsymbol{c}|) = S^{\rho}_{max}(\boldsymbol{w}, |\boldsymbol{w}|)$ as shown above.

Adaptive sharpness for ViTs. A typical MLP block of ViTs contains the following operations: `LN`→`Linear` →`GELU`→`Linear` where `LN` denotes LayerNorm, and pre-softmax self-attention weights are computed as $ZW_QW_K^\top Z^\top$ where $Z \in \mathbb{R}^{P \times D}$ is the matrix of P D-dimensional tokens. The network thus has the following invariances to multiplication/division by α: (1) between `LN` and `Linear` in MLP, (2) between W_Q in W_K in self-attention, (3) between two `Linear` layers that have GELU in-between for which $\text{GELU}(\alpha z)/\alpha \approx \text{GELU}(z)$. Moreover, at the beginning of the network there is a part of the network which is invariant to the scale of the `Linear` layer (`Linear`→`LN`). Similarly to ResNets, all these invariances are multiplicative, so the argument about the invariance of elementwise adaptive sharpness is the same.

Scale-sensitivity for classification losses. However, adaptive sharpness remains sensitive to the *scale* of the classifier, meaning that the sharpness together with the cross-entropy loss keep decreasing to zero after reaching zero training error. This can be seen even for linear models for which scaling the weight vector by a constant changes the adaptive sharpness as shown in Fig. 7.1. To fix this issue, Tsuzuku et al. (2020) propose to use normalization of the logits $f_{\boldsymbol{w}}$, i.e.:

$$\tilde{f}_{\boldsymbol{w}}(\boldsymbol{x}) \triangleq \frac{f_{\boldsymbol{w}}(\boldsymbol{x})}{\sqrt{\frac{1}{K}\sum_{i=1}^{K}(f_{\boldsymbol{w}}(\boldsymbol{x})_i - f_{avg}(\boldsymbol{x}))^2}}, \tag{7.5}$$

where $f_{avg}(\boldsymbol{x}) = \frac{1}{K}\sum_{j=1}^{K} f_{\boldsymbol{w}}(\boldsymbol{x})_j$. This provably fixes the scaling issue meaning that scaling the output layer by $\alpha \in \mathbb{R}_+$ does not affect the logits. Moreover, this change can make models having different training loss more comparable to each other.

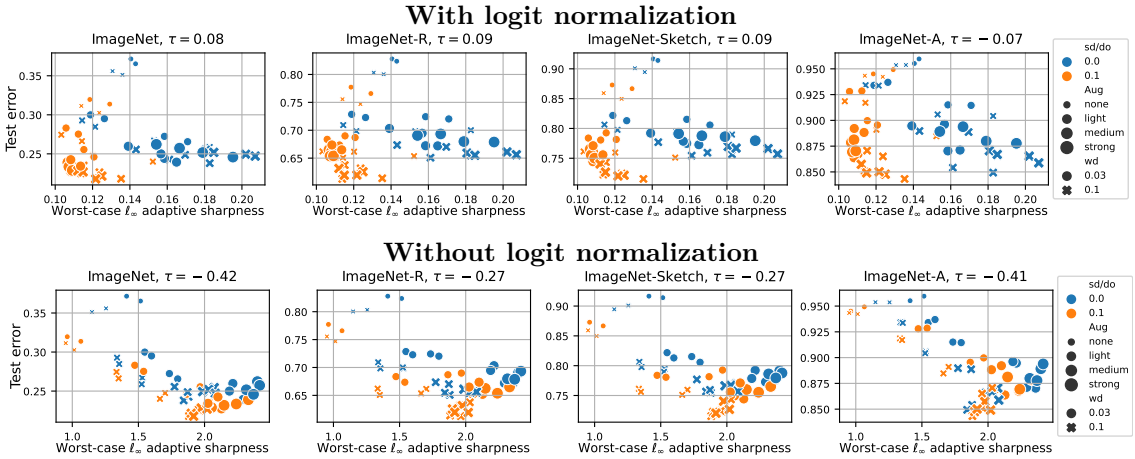


Figure 7.2: ViT-B/16 trained from scratch on ImageNet-1k. We show for 56 models from Steiner et al. (2021) the test error on ImageNet and its OOD variants vs. worst-case ℓ_∞ sharpness with (top) or without (bottom) normalization at $\rho = 0.002$. The color indicates models trained with stochastic depth (sd) and dropout (do), markers and their size indicate the strength of weight decay (wd) and augmentations (aug), and τ indicates the rank correlation coefficient from Eq. (7.4). Overall, the correlation of sharpness with test error is either close to zero or even negative.

7.4.3 How to Compute Worst-Case Sharpness Efficiently?

Estimation of worst-case sharpness involves solving a constrained maximization problem typically using projected gradient ascent which can be sensitive to its hyperparameters, primarily the step size. To avoid doing extensive grid searches over the hyperparameters of gradient ascent for each model, we choose to use *Auto-PGD* (Croce and Hein, 2020b) (see Algorithm 5 in Appendix for the precise formulation). Auto-PGD is a *hyperparameter-free* method designed to accurately estimate adversarial robustness by solving a similar optimization problem to worst-case sharpness but over the input space instead of the parameter space. As in ℓ_∞ and ℓ_2 versions of Auto-PGD, for each gradient step, we use gradient-sign and plain-gradient updates, respectively, but we make them proportional to $|\boldsymbol{w}|$, to better take into account the geometry induced by elementwise adaptive sharpness. We show in Sec. 7.15.2 in Appendix that as few as 20 steps are typically sufficient to converge with Auto-PGD.

7.5 Sharpness vs. Generalization: Modern Setup

The current understanding of the relationship between sharpness and generalization is based on experiments on non-residual convolution networks and small datasets like CIFAR-10 and SVHN (Jiang et al., 2019). We revisit here this relationship for state-of-the-art transformers trained from scratch on ImageNet-1k and CLIP / BERT fine-tuned on ImageNet-1k / MNLI. We explore both in-distribution (ID) and out-of-distribution (OOD) generalization due to the common intuition that flatter models are expected to be more robust (Cha et al., 2021). We focus on worst-case ℓ_∞ adaptive sharpness with low m

(256) since it appears to be one of the most promising sharpness definitions (Kwon et al., 2021). We compute sharpness with and without logit normalization, and provide *average-case* sharpness for different radii ρ in Appendix. We focus primarily on the relationship between sharpness and *test error* but we also discuss sharpness vs. *generalization gap* in Sec. 7.9 in Appendix.

Training on ImageNet-1k from scratch. To investigate the relationship between sharpness and generalization for large-scale settings, we evaluate ViT models from Steiner et al. (2021), using ViT-B/16-224 weights. Those were trained from scratch on ImageNet-1k for 300 epochs with different hyperparameter settings, and subsequently fine-tuned on the same dataset for 20.000 steps with 2 different learning rates. The different hyperparameters include augmentations, weight decay, and stochastic depth / dropout, leading to a rich pool of 56 models with test errors ranging from 21.8% to 37.2%. As shown in Figure 7.2 (first column), neither the sharpness measure computed with nor without logit normalization can effectively distinguish model performance. Logit-normalized sharpness effectively separates models with stochastic depth / dropout (sd/do from now on) from those without by grouping them into two distinct clusters (blue and orange). However, these clusters do not correspond to a separation by test error. For the OOD tasks (ImageNet-R, ImageNet-Sketch, ImageNet-A), within each cluster, the models trained with higher weight decay yield lower test error fairly consistently. However, this ranking is not captured by sharpness, which only disentangles the sd/do clusters. For sharpness without logit normalization, the sd/do clusters are not well-separated. Surprisingly, there is a consistent *negative* correlation between sharpness and test error, both on ID and OOD data, i.e. the flattest models tend to have the largest test error. Evaluation for other radii, average-case sharpness measures (App. 7.10) and for ViTs pretrained on IN-21k and fine-tuned on IN-1k (App. 7.11) similarly suggest that sharpness does not consistently capture generalization properties. When considering IN-1k and IN-21k pre-trained models together (App. 7.12) we even find similar or *higher* sharpness for significantly better-generalizing models. Then, for none of the settings studied, we can confirm either the strong or weak hypotheses.

Fine-tuning on ImageNet-1k from CLIP. We investigate fine-tuning from CLIP (Radford et al., 2021), which is a crucial approach due to the popularity of CLIP features (Ramesh et al., 2022), its fast training time, and its ability to achieve higher accuracy. We study the pool of classifiers obtained by Wortsman et al. (2022a) who fine-tuned a CLIP ViT-B/32 model on ImageNet multiple times by randomly selecting training hyperparameters such as learning rate, number of epochs, weight decay, label smoothing and augmentations. This set of 71 fine-tuned models, along with the base model, allows us to study how well generalization and training hyperparameters are captured by sharpness. The leftmost column of Fig. 7.3 illustrates that worst-case ℓ_∞ adaptive sharpness does not effectively predict which classifiers have the lowest test error on ImageNet. Furthermore, there is a consistent negative correlation between sharpness and test error when evaluating classifiers on the distribution shifts ImageNet-R (Hendrycks et al., 2021b), ImageNet-Sketch (Wang et al., 2019a) and ImageNet-A (Hendrycks et al., 2021c) (second to fourth columns). We further

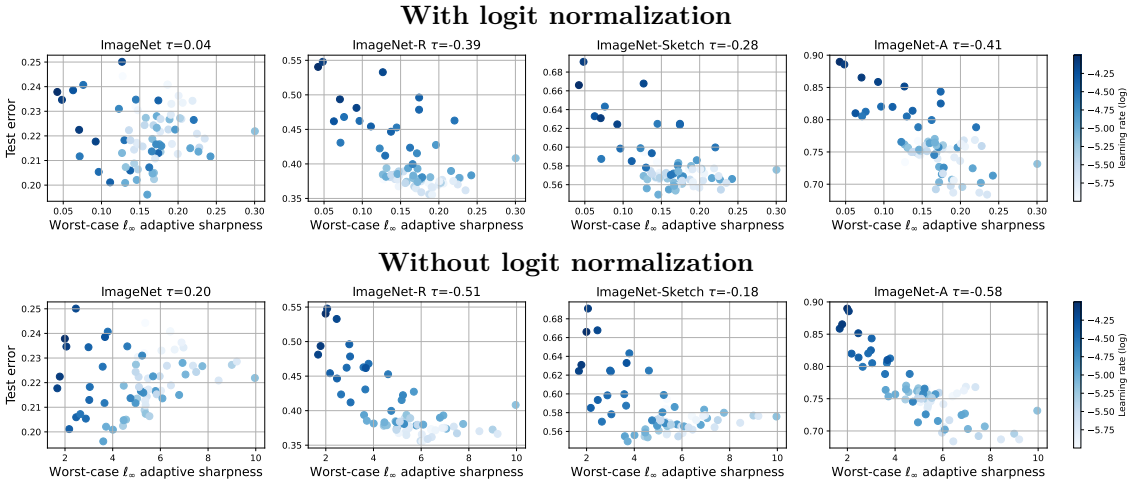


Figure 7.3: Fine-tuning CLIP ViT-B/32 on ImageNet-1k. We show for 72 models from Wortsman et al. (2022a) the test error on ImageNet or its variants (distribution shifts) vs worst-case ℓ_∞ sharpness with (top) or without (bottom) normalization at $\rho = 0.002$. Darker color indicates larger learning rate used for fine-tuning.

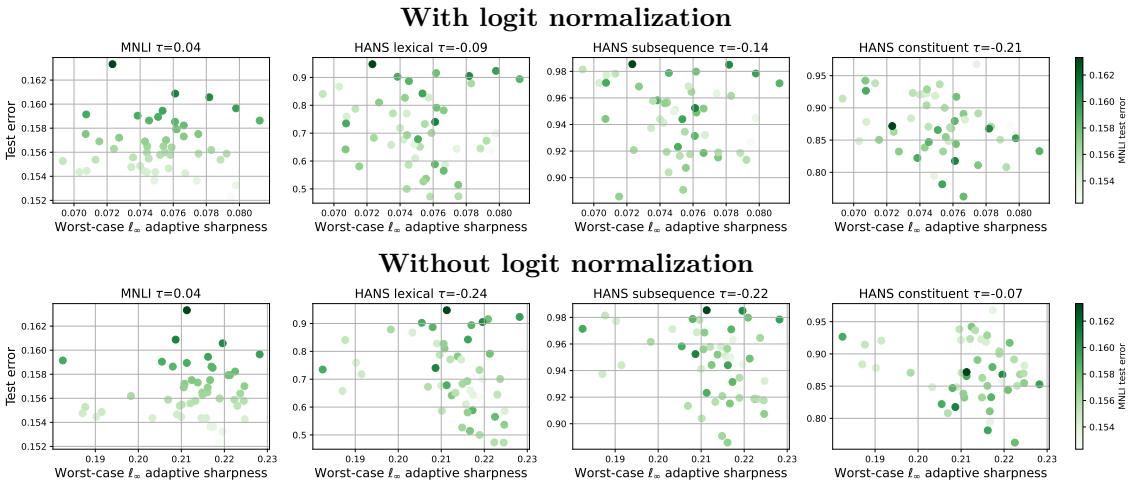


Figure 7.4: Fine-tuning BERT on MNLI. We show for 50 models the error on MNLI or out-of-distribution domains (HANS subsets) vs worst-case ℓ_∞ sharpness with (top) or without (bottom) normalization at $\rho = 0.0005$. Darker color indicates higher test error on MNLI.

notice that, in contrast with ImageNet, higher test errors on these datasets go in parallel with higher learning rates used for fine-tuning (darker color in the plots). Indeed, smaller learning rates lead to smaller changes in the features of the base CLIP model which are more robust to distribution shifts since they were obtained from a much larger dataset than ImageNet. Finally, similar observations hold for the other sharpness definition and radii (App. 7.13).

Fine-tuning on MNLI from BERT. We explore fine-tuning from BERT (Devlin et al., 2019), to expand our analysis beyond vision tasks. To study the linguistic generalization of multiple classifiers trained on the same dataset, McCoy et al. (2020) have fine-

tuned BERT 100 times on the Multi-genre Natural Language Inference (MNLI) dataset (Williams et al., 2018) varying exclusively the random seed across runs. These random seeds affect the initialization of the classifier and the scanning order of the training data for SGD. All these classifiers achieve very similar in-distribution generalization, i.e. on MNLI test points, but behave differently on the out-of-distribution tasks represented by the HANS dataset (McCoy et al., 2019). For example, in one of HANS sub-domains the accuracy of the models ranges from 5% to 55%. We randomly choose 50 of the 100 available classifiers, and compute the different measures of sharpness for various radii. Fig. 7.4 shows how the worst-case ℓ_∞ adaptive sharpness, with and without logit normalization, correlates with test error on MNLI and three HANS tasks. We observe that the correlation is weak and does not exceed 0.04, even for datasets like HANS lexical (second column) where test errors vary significantly (between 45% and 95%). Moreover, in some cases the correlation is weakly negative suggesting that on average sharper models tend to generalize slightly better. Results for other radii can be found in App. 7.14.

Summary of the findings. To conclude, *none* of the settings studied above support either the strong or weak hypotheses about the role of sharpness. Contrary to our expectations, CLIP models fine-tuned on ImageNet suggest that flatter solutions consistently generalize *worse* on OOD data. Finally, sharpness is not useful to distinguish different solutions found by fine-tuning BERT on MNLI. All this evidence suggests that the intuitive ideas about the generalization benefits of flat minima are *not supported in the modern settings*.

7.6 Why Doesn't Sharpness Correlate Well with Generalization?

The goal of this section is to clarify the disconnect between sharpness and generalization in the modern setup. We first revisit sharpness in a controlled environment on CIFAR-10, then explore the different sharpness definitions for a simple model where generalization is well understood.

7.6.1 The Role of Sharpness in a Controlled Setup

Motivation. We consider three potential explanations for why sharpness does not correlate well with generalization in the previous section: (1) the use of transformers instead of typical convolutional networks, (2) the use of much larger datasets (ImageNet vs. CIFAR-10), (3) the need to measure sharpness closer to a global minimum. We thus train 200 ResNets-18 and 200 ViTs on CIFAR-10 in a setting similar to Jiang et al. (2019) and Kwon et al. (2021), and evaluate sharpness only for models that reach *at most* 1% *training error*. This is in contrast to the ImageNet models from the previous section that are not necessarily trained to $\approx 0\%$ training error as it is usually not necessary in practice. Being closer to a global minimum ensures that the worst-case sharpness captures more

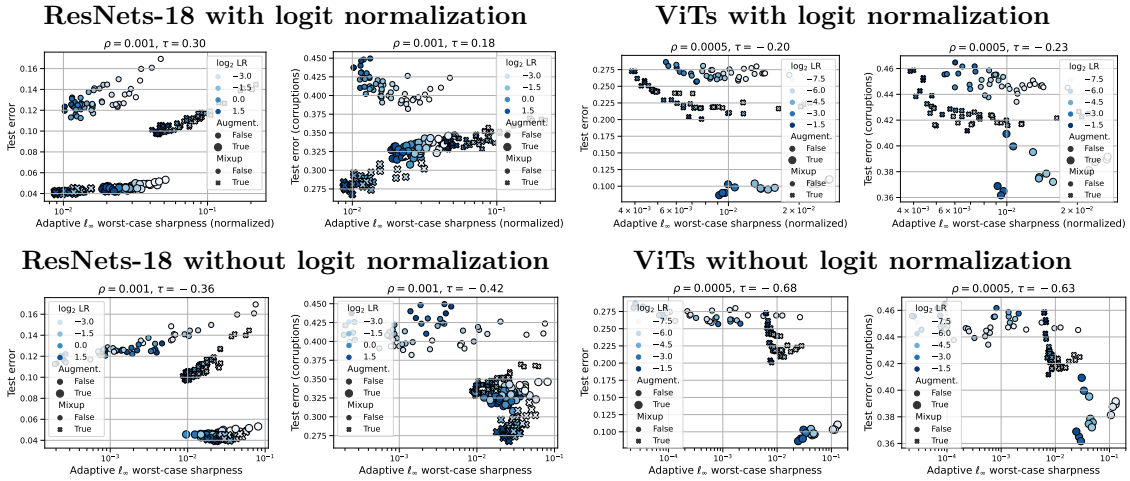


Figure 7.5: Training from scratch on CIFAR-10. Normalized and unnormalized ℓ_∞ adaptive sharpness vs. standard and OOD test error on common corruptions for ResNets-18 and ViTs. For other sharpness definitions (ℓ_2/ℓ_∞, average-/worst-case, etc) and multiple sharpness radii ρ, see App. 7.15.4.

the curvature by preventing first-order terms from dominating in Eq. 7.3.

Setup. We train models for 200 epochs using SGD with momentum and linearly decreasing learning rates after a linear warm-up for the first 40% iterations. We use the SimpleViT architecture from the `vit-pytorch` library which is a modification of the standard ViT (Dosovitskiy et al., 2021) with a fixed positional embedding and global average pooling instead of the CLS embedding. We vary the learning rate, $\rho \in \{0, 0.05, 0.1\}$ of SAM (Foret et al., 2021), mixup ($\alpha = 0.5$) (Zhang et al., 2017b), and standard augmentations combined with RandAugment (Cubuk et al., 2020). We only show models that have $\leq 1\%$ training error.

Observations. We benchmark 12 different sharpness definitions: ℓ_2 vs. ℓ_∞, average- vs. worst-case, standard vs. adaptive, with vs. without logit normalization, and consider different perturbation radii ρ. We report most of these results in App. 7.15 and here highlight only ℓ_∞ adaptive sharpness in Fig. 7.5. We observe that for ResNets, there is a strong correlation between sharpness and test error but *only within each subgroup of training parameters* such as augmentations and mixup. Importantly, sharpness does not correctly capture generalization between different subgroups leading to low positive or negative correlation (0.30 and -0.36). For ViTs, we do not observe strong positive correlation even within each subgroup (in fact, without logit normalization the correlation is noticeably negative -0.68), and many models with an order of magnitude difference in sharpness can have the same test error. Moreover, we do not consistently observe that models with the lowest sharpness generalize best. For OOD generalization on common image corruptions (Hendrycks and Dietterich, 2019), the trend is even less clear and the subgroups are mixed. We note that similar conclusions hold for other sharpness radii ρ and definitions which we show in App. 7.15.4. Moreover, in App. 7.15 we also analyze the role of data points used to evaluate sharpness (with and without augmentations),

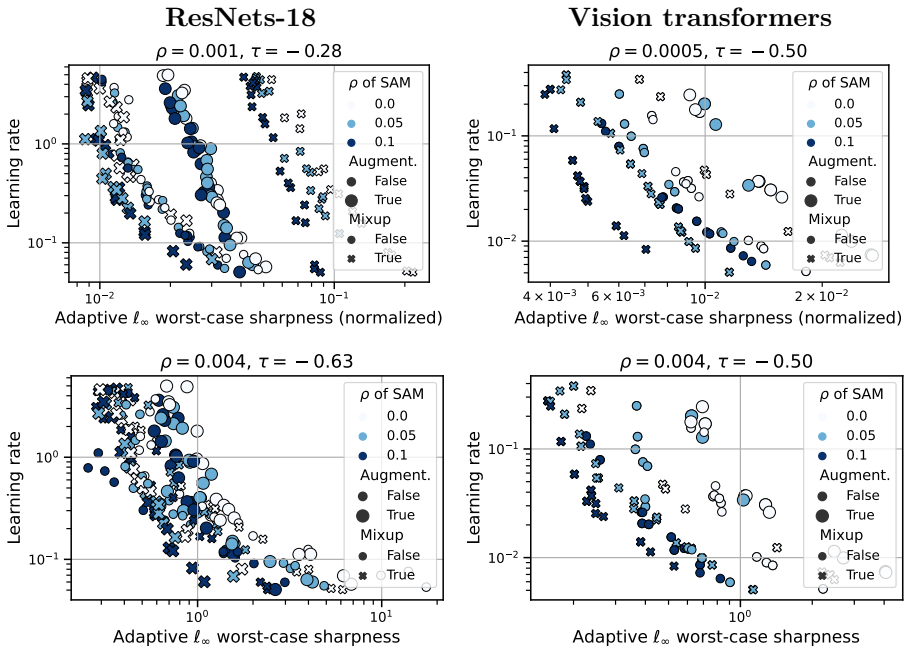


Figure 7.6: Training from scratch on CIFAR-10. Sharpness negatively correlates with the *learning rate*, especially within each subgroup defined by the same values of `augment` × `mixup`.

number of iterations of Auto-PGD for worst-case sharpness, and different m in worst-case m-sharpness (Foret et al., 2021). In conclusion, even in this controlled small-scale setup that includes more established architectures like ResNets, we find no empirical support to either the strong or weak hypothesis.

Sharpness captures the learning rate even when it is not helpful to predict generalization. Prior works have shown a robust link between the learning rate of first-order methods and standard sharpness definitions such as $\lambda_{\max}(\nabla^2 L(\boldsymbol{w}))$ and $\operatorname{tr}(\nabla^2 L(\boldsymbol{w}))$ (Cohen et al., 2021; Wu et al., 2022). However, the connection between the learning rate and *adaptive* sharpness remains elusive, so we investigate it empirically in Fig. 7.6. For both ResNets and ViTs, we observe a significant negative correlation, especially within each subgroup defined by the same values of `augment` × `mixup`. This is however *not* always a desirable property for predicting generalization. On the one hand, monotonically capturing the learning rates can be useful in setting like training ResNets from scratch (Li et al., 2019d). On the other hand, large learning rates do not preserve the original features and can significantly harm OOD generalization for fine-tuning (Wortsman et al., 2022b). We also see a negative correlation between sharpness and learning rate for CLIP models fine-tuned on ImageNet in Fig. 7.20, shown in App. 7.13. However, for these models, we do not have subgroups as clearly defined as for the CIFAR-10 models so we cannot see a more fine-grained trend. Finally, we note that whenever learning rates have a beneficial regularization effect, it is closely tied to the amount of stochastic noise in SGD (Jastrzebski et al., 2017; Andriushchenko et al., 2023d). This amount is equally determined by other hyperparameters like batch size, momentum coefficient, or weight decay for normalized networks (see Li et al. (2020b) for a discussion on the intrinsic learning rate). These

parameters are commonly varied in studies on sharpness vs. generalization (Jiang et al., 2019; Kwon et al., 2021; Bisla et al., 2022) but all reflect essentially the same underlying trend.

7.6.2 Is Sharpness the Right Quantity in the First Place? Insights from Simple Models

Here, we study the link between sharpness and generalization for sparse regression with *diagonal linear networks* for which the ℓ_1 norm of the solution is predictive of generalization. This simple model suggests that sharpness measures which are universally correlated with better generalization across all possible data distributions simply do not exist.

Diagonal linear networks are defined as predictors $\langle \boldsymbol{x}, \boldsymbol{\beta} \rangle$ with parameterization $\boldsymbol{\beta} = \boldsymbol{u} \odot \boldsymbol{v}$ for weights $\boldsymbol{w} = \left[\begin{smallmatrix} \boldsymbol{u} \\ \boldsymbol{v} \end{smallmatrix}\right] \in \mathbb{R}^{2d}$. They have been widely studied as the simplest non-trivial neural network (Woodworth et al., 2020; Pesme et al., 2021). We consider an overparametrized sparse regression problem for a data matrix $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ and label vector $\boldsymbol{y}$:

$$L(\boldsymbol{w}) := \|\boldsymbol{X}(\boldsymbol{u} \odot \boldsymbol{v}) - \boldsymbol{y}\|_2^2, \tag{7.6}$$

for which the ground truth $\boldsymbol{\beta}^*$ is a sparse vector (i.e., most coordinates are zeros) and there exist many solutions $\boldsymbol{w}$ such that $L(\boldsymbol{w}) = 0$. Assuming whitened data $\boldsymbol{X}^\top \boldsymbol{X} = \boldsymbol{I}$ and that $\boldsymbol{w}$ is a global minimum, the Hessian of the loss L simplifies to

$$\nabla^2 L(\boldsymbol{w}) = \begin{bmatrix} \operatorname{diag}(\boldsymbol{v} \odot \boldsymbol{v}) & \operatorname{diag}(\boldsymbol{u} \odot \boldsymbol{v}) \\ \operatorname{diag}(\boldsymbol{u} \odot \boldsymbol{v}) & \operatorname{diag}(\boldsymbol{u} \odot \boldsymbol{u}) \end{bmatrix}.$$

We first consider standard definitions of *local* (i.e., $\rho \to 0$) sharpness for which we have a closed-form expression. The average-case local sharpness is equal to $\operatorname{tr}(\nabla^2 L(\boldsymbol{w})) = \sum_{i=1}^d u_i^2 + v_i^2$ while the worst-case local sharpness at a minimum is $\lambda_{\max}(\nabla^2 L(\boldsymbol{w})) = \max_{1 \le i \le d} v_i^2 + u_i^2$ (see Sec. 7.8.2 for details). Importantly, both average- and worst-case local sharpness are not invariant under α-reparametrization $(\alpha \boldsymbol{u}, \boldsymbol{v}/\alpha)$ while the predictor $\boldsymbol{\beta} = \boldsymbol{u} \odot \boldsymbol{v}$ is. This fact emphasizes the need for a measure of the sharpness that adjusts to the changing scale of the parameters as the adaptive sharpness. Indeed, with the carefully selected elementwise scaling $c_i = \sqrt{|v_i|/|u_i|}$ for $1 \le i \le d$ and $c_i = \sqrt{|u_i|/|v_i|}$ for $d < i \le 2d$, we obtain for the average-case and worst-case adaptive local sharpness

$$\begin{aligned} S^\rho_{avg}(\boldsymbol{w}, \boldsymbol{c}) &= \frac{1}{2}\sum_{i=1}^d u_i^2 |v_i|/|u_i| + \frac{1}{2}\sum_{i=1}^d v_i^2 |u_i|/|v_i| = \|\boldsymbol{\beta}\|_1, \\ S^\rho_{max}(\boldsymbol{w}, \boldsymbol{c}) &= \max_{1 \le i \le d} |u_i||v_i| = \|\boldsymbol{\beta}\|_\infty. \end{aligned}$$

We first note that both definitions of adaptive sharpness are invariant under α-reparametrization as they only depend on the predictor $\boldsymbol{\beta}$. However, average and worst-case sharpness do not capture the same properties of $\boldsymbol{\beta}$. In particular, $\|\boldsymbol{\beta}\|_1$ is a generalization measure

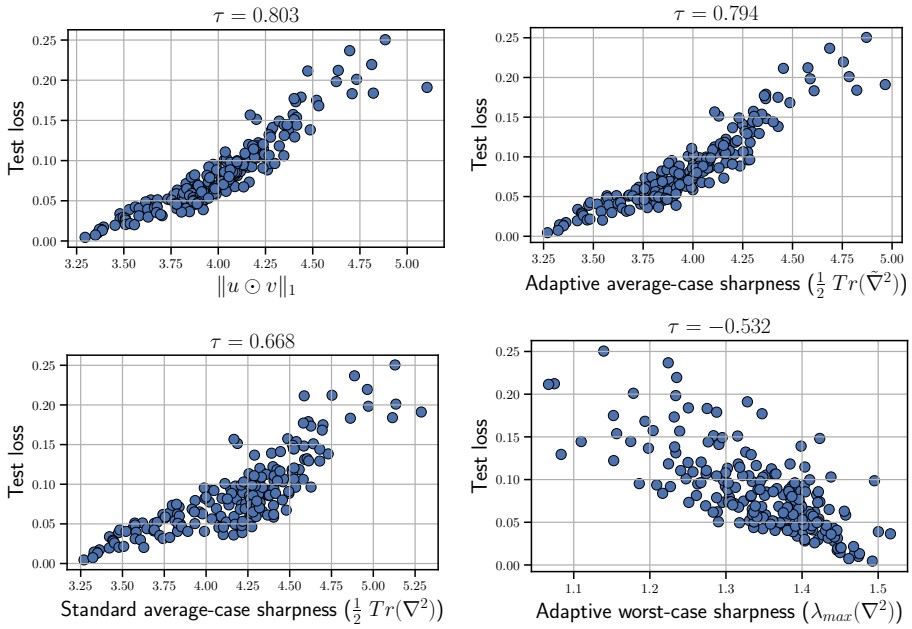


Figure 7.7: Different generalization measures for diagonal linear networks. $\tilde{\nabla}^2$ denotes the rescaled Hessian corresponding to adaptive sharpness.

that correctly captures the sparsity of the linear predictor which is a good indicator of generalization for a *sparse* $\boldsymbol{\beta}^*$. In contrast, $\|\boldsymbol{\beta}\|_\infty$ is a generalization measure that is more suitable to capture how uniform the weights of $\boldsymbol{\beta}$ are which is a good predictor of generalization for a *dense* $\boldsymbol{\beta}^*$. Finally, we note that using $\boldsymbol{c} = \boldsymbol{w}$ in adaptive sharpness would instead lead to $\|\boldsymbol{\beta}\|_2^2$ and $\|\boldsymbol{\beta}\|_\infty^2$ that would have a different interpretation. This simple model highlights that the sharpness definition that correlates well with generalization is data-dependent and in general S_{avg} and S_{max} capture very different trends.

To further illustrate this point, we train 200 diagonal linear networks to 10^{-5} training loss on a sparse regression task ($d = 200$ with 90% sparsity) with different learning rates and random initializations. We show the results in Fig. 7.7 which illustrate that (1) $\|\boldsymbol{u} \odot \boldsymbol{v}\|_1$ is approximated well by $\frac{1}{2}\text{tr}(\tilde{\nabla}^2 L(\boldsymbol{w}))$, (2) $\text{tr}(\tilde{\nabla}^2 L(\boldsymbol{w}))$ correlates better than $\text{tr}(\nabla^2 L(\boldsymbol{w}))$ so the adaptive part is important, (3) the relationship between $\text{tr}(\tilde{\nabla}^2 L(\boldsymbol{w}))$ and $\lambda_{\max}(\tilde{\nabla}^2 L(\boldsymbol{w}))$ can be even reverse showing that different sharpness definitions capture totally different trends. We also note that even with the right definition of sharpness, the correlation is not perfect (around $\tau = 0.8$) and there is always some non-negligible gap in predicting the test loss. Overall, we conclude that finding a sharpness definition that correlates well with generalization requires understanding both the role of the data distribution and its interaction with the architecture. It is possible in very simple cases but appears extremely challenging for complex architectures like vision transformers on complex real-world datasets like ImageNet.

7.7 Conclusions

Our results suggest that even reparametrization-invariant sharpness is *not* a good indicator of generalization in the modern setting. While there definitely exist restricted settings where correlation between sharpness and generalization is significantly positive (e.g., for ResNets on CIFAR-10 with a specific combination of augmentations and mixup), it is not true anymore when we compare all models *jointly*. Moreover, the correlation, even within subgroups of models defined by augmentations, is much lower for vision transformers. Thus, we believe it is important to rethink the intuitive understanding of sharpness based on the geometric intuition about the shift of the loss surface. Moreover, our findings suggest that one should avoid blanket statements like *"flatter minima generalize better"* since even when they are only intended to imply *correlation*, their correctness still depends on a number of factors such as data distribution, model family, or initialization schemes (i.e., random vs. from pretrained weights).

Appendix

The appendix is organized as follows:

- Sec. 7.8: omitted derivations for sharpness when $\rho \to 0$, first for the general case and then specifically for diagonal linear networks.
- Sec. 7.9: figures with correlation between sharpness and *generalization gap*. We observe a similar trend between sharpness and *generalization gap* as between sharpness and *test error* which is reported in the main part.
- Sec. 7.10: additional figures about ViTs from Steiner et al. (2021) trained with different hyperparameter settings on ImageNet-1k. We observe that different sharpness variants are not predictive of the performance on ImageNet and the OOD datasets, typically only separating models by stochastic depth / dropout, but not ranking them according to generalization, and often even yielding a negative correlation with OOD test error.
- Sec. 7.11: figures about ViTs from Steiner et al. (2021) pre-trained on ImageNet-21k and then fine-tuned on ImageNet-1k. The observations are very similar to those for training on ImageNet-1k from scratch: sharpness variants are not predictive of the performance on ImageNet, and they often lead to a negative correlation with OOD test error.
- Sec. 7.12: figures for combined analysis of ViTs from Steiner et al. (2021) both with and without ImageNet-21k pre-training. We find the better-generalizing models pretrained on ImageNet-21k to have significantly higher worst-case sharpness and roughly equal or higher logit-normalized average-case adaptive sharpness, underlining that the models' generalization properties resulting from different pretraining datasets are not captured.
- Sec. 7.13: additional details and figures for CLIP models fine-tuned on ImageNet. We observe that sharpness variants are not predictive of the performance on ImageNet and ImageNet-V2. Moreover, there is in most cases a negative correlation with test error in presence of distribution shifts which is likely to be related to the influence that the learning rate has on sharpness.
- Sec. 7.14: additional details and figures for BERT models fine-tuned on MNLI. We find that all sharpness variants we consider are not predictive of the generalization performance of the model, and in some cases there is rather a weak negative correlation between sharpness and test error on out-of-distribution tasks from HANS.
- Sec. 7.15: additional details and ablation studies for CIFAR-10 models. We analyze the role of data used to evaluate sharpness, the role of the number of iterations

in Auto-PGD, the role of m in m-sharpness, and the influence of different sharpness definitions and radii on correlation with generalization. Overall, we conclude that none of the considered sharpness definitions or radii correlates positively with generalization nor that low sharpness implies good performance of the model.

Also, for the sake of convenience, we provide in Table 7.1, Table 7.2, Table 7.3, and Table 7.4 a summary of correlation coefficients τ between sharpness and generalization for all our experiments (except ablation studies).

Table 7.1: A summary of correlation between sharpness and generalization for all experiments on **ImageNet**. We boldface entries with $|\tau| > 0.5$ suggesting a reasonably strong correlation. LogitNorm stands for *logit normalization* and IN stands for *ImageNet*.

ImageNet-1k models trained from scratch

			Rank correlation coefficient τ					
Sharpness	**LogitNorm**	ρ	**IN**	**IN-v2**	**IN-R**	**IN-Sketch**	**IN-A**	**ObjectNet**
Worst-case ℓ_∞	Yes	0.001	0.09	0.08	0.10	0.10	-0.06	0.04
Worst-case ℓ_∞	Yes	0.002	0.08	0.08	0.09	0.09	-0.07	0.03
Worst-case ℓ_∞	Yes	0.004	-0.11	-0.11	-0.06	-0.06	-0.23	-0.16
Worst-case ℓ_∞	No	0.001	-0.42	-0.43	-0.27	-0.28	-0.45	-0.45
Worst-case ℓ_∞	No	0.002	-0.42	-0.42	-0.27	-0.27	-0.41	-0.45
Worst-case ℓ_∞	No	0.004	-0.34	-0.34	-0.20	-0.20	-0.36	-0.36
Avg-case ℓ_∞	Yes	0.05	0.46	0.44	0.38	0.42	0.31	0.39
Avg-case ℓ_∞	Yes	0.1	0.44	0.43	0.39	0.43	0.29	0.39
Avg-case ℓ_∞	Yes	0.2	0.42	0.42	0.39	0.42	0.29	0.38
Avg-case ℓ_∞	No	0.05	**-0.55**	**-0.56**	-0.40	-0.42	**-0.57**	**-0.60**
Avg-case ℓ_∞	No	0.1	-0.44	-0.43	-0.28	-0.32	-0.47	-0.47
Avg-case ℓ_∞	No	0.2	0.13	0.15	0.26	0.23	0.05	0.11

ImageNet-1k models fine-tuned from IN-21k

			Rank correlation coefficient τ					
Sharpness	**LogitNorm**	ρ	**IN**	**IN-v2**	**IN-R**	**IN-Sketch**	**IN-A**	**ObjectNet**
Worst-case ℓ_∞	Yes	0.001	-0.49	-0.49	-0.44	-0.33	**-0.53**	-0.46
Worst-case ℓ_∞	Yes	0.002	-0.48	-0.48	-0.46	-0.33	**-0.51**	-0.44
Worst-case ℓ_∞	Yes	0.004	-0.45	-0.43	-0.41	-0.33	-0.45	-0.42
Worst-case ℓ_∞	No	0.001	-0.13	-0.09	-0.05	0.05	-0.13	-0.09
Worst-case ℓ_∞	No	0.002	-0.10	-0.03	-0.01	0.11	-0.07	-0.02
Worst-case ℓ_∞	No	0.004	-0.10	-0.01	-0.01	0.11	-0.06	0.00
Avg-case ℓ_∞	Yes	0.05	-0.11	-0.08	-0.11	-0.07	-0.06	-0.06
Avg-case ℓ_∞	Yes	0.1	-0.12	-0.11	-0.14	-0.10	-0.09	-0.08
Avg-case ℓ_∞	Yes	0.2	-0.25	-0.24	-0.25	-0.23	-0.25	-0.24
Avg-case ℓ_∞	No	0.05	-0.02	-0.04	-0.03	-0.02	-0.05	-0.06
Avg-case ℓ_∞	No	0.1	-0.07	-0.10	-0.08	-0.08	-0.11	-0.10
Avg-case ℓ_∞	No	0.2	-0.11	-0.11	-0.10	-0.11	-0.12	-0.13

ImageNet-1k models fine-tuned from CLIP

			Rank correlation coefficient τ					
Sharpness	**LogitNorm**	ρ	**IN**	**IN-v2**	**IN-R**	**IN-Sketch**	**IN-A**	**ObjectNet**
Worst-case ℓ_∞	Yes	0.001	-0.04	-0.16	-0.23	-0.26	-0.25	-0.36
Worst-case ℓ_∞	Yes	0.002	0.04	-0.10	-0.39	-0.28	-0.41	-0.47
Worst-case ℓ_∞	Yes	0.004	-0.08	-0.19	-0.12	-0.16	-0.17	-0.27
Worst-case ℓ_∞	No	0.001	0.19	0.09	-0.37	-0.06	**-0.57**	-0.48
Worst-case ℓ_∞	No	0.002	0.20	0.08	**-0.51**	-0.18	**-0.58**	**-0.51**
Worst-case ℓ_∞	No	0.004	0.02	-0.05	**-0.51**	-0.27	-0.45	-0.33
Avg-case ℓ_∞	Yes	0.001	-0.03	-0.18	-0.36	-0.34	-0.33	-0.46
Avg-case ℓ_∞	Yes	0.002	-0.21	-0.32	-0.02	-0.27	-0.06	-0.21
Avg-case ℓ_∞	Yes	0.004	-0.19	-0.21	0.26	-0.03	0.23	0.06
Avg-case ℓ_∞	No	0.001	0.13	-0.01	**-0.62**	-0.26	**-0.67**	**-0.60**
Avg-case ℓ_∞	No	0.002	0.06	0.03	-0.34	-0.12	**-0.50**	-0.37
Avg-case ℓ_∞	No	0.004	0.19	0.21	-0.12	0.09	-0.21	-0.08

Table 7.2: A summary of correlation between sharpness and generalization for all experiments on **MNLI** for models fine-tuned from BERT. We boldface entries with $|\tau| > 0.5$ suggesting a reasonably strong correlation. LogitNorm stands for *logit normalization*.

MNLI models fine-tuned from BERT						
			Rank correlation coefficient τ			
Sharpness	**LogitNorm**	ρ	**MNLI**	**HANS-L**	**HANS-S**	**HANS-C**
Worst-case ℓ_∞	Yes	0.0005	0.04	-0.09	-0.14	-0.21
Worst-case ℓ_∞	Yes	0.001	-0.09	-0.09	-0.13	-0.18
Worst-case ℓ_∞	Yes	0.002	0.05	-0.09	-0.14	-0.17
Worst-case ℓ_∞	No	0.0005	0.04	-0.24	-0.22	-0.07
Worst-case ℓ_∞	No	0.001	0.04	-0.13	-0.15	-0.15
Worst-case ℓ_∞	No	0.002	-0.11	-0.15	-0.12	-0.13
Avg-case ℓ_∞	Yes	0.1	-0.35	-0.46	-0.28	0.17
Avg-case ℓ_∞	Yes	0.2	-0.37	-0.48	-0.28	0.24
Avg-case ℓ_∞	Yes	0.4	0.01	-0.29	-0.27	0.05
Avg-case ℓ_∞	No	0.1	-0.34	-0.31	-0.23	0.13
Avg-case ℓ_∞	No	0.2	-0.34	**-0.58**	-0.39	0.16
Avg-case ℓ_∞	No	0.4	0.04	-0.16	-0.09	0.05

Table 7.3: A summary of correlation between sharpness and generalization for all experiments on **CIFAR-10** for ResNets-18 trained from scratch. We boldface entries with $|\tau| > 0.5$ suggesting a reasonably strong correlation. LogitNorm stands for *logit normalization*.

ResNets-18 trained from scratch on CIFAR-10

			Rank correlation coefficient τ	
Sharpness	**LogitNorm**	ρ	**CIFAR-10**	**CIFAR-10-C**
Standard avg-case ℓ_2	No	0.05	0.14	0.04
Standard avg-case ℓ_2	No	0.1	0.26	0.19
Standard avg-case ℓ_2	No	0.2	0.28	0.21
Standard avg-case ℓ_2	No	0.4	0.28	0.20
Standard worst-case ℓ_2	No	0.25	0.17	0.10
Standard worst-case ℓ_2	No	0.5	0.24	0.16
Standard worst-case ℓ_2	No	1.0	0.25	0.18
Standard worst-case ℓ_2	No	2.0	0.22	0.14
Adaptive avg-case ℓ_2	No	0.05	-0.37	-0.46
Adaptive avg-case ℓ_2	No	0.1	**-0.50**	**-0.53**
Adaptive avg-case ℓ_2	No	0.2	-0.42	-0.41
Adaptive avg-case ℓ_2	No	0.4	-0.31	-0.31
Adaptive worst-case ℓ_2	No	0.25	-0.36	-0.39
Adaptive worst-case ℓ_2	No	0.5	-0.42	-0.36
Adaptive worst-case ℓ_2	No	1.0	-0.27	-0.17
Adaptive worst-case ℓ_2	No	2.0	-0.17	-0.07
Adaptive avg-case ℓ_2	Yes	0.05	0.18	0.07
Adaptive avg-case ℓ_2	Yes	0.1	0.07	-0.04
Adaptive avg-case ℓ_2	Yes	0.2	-0.14	-0.26
Adaptive avg-case ℓ_2	Yes	0.4	-0.43	-0.58
Adaptive worst-case ℓ_2	Yes	0.25	0.19	0.14
Adaptive worst-case ℓ_2	Yes	0.5	0.07	0.00
Adaptive worst-case ℓ_2	Yes	1.0	-0.13	-0.22
Adaptive worst-case ℓ_2	Yes	2.0	**-0.52**	**-0.58**
Standard avg-case ℓ_∞	No	0.1	0.16	0.08
Standard avg-case ℓ_∞	No	0.2	0.28	0.21
Standard avg-case ℓ_∞	No	0.4	0.28	0.20
Standard avg-case ℓ_∞	No	0.8	0.28	0.20
Standard worst-case ℓ_∞	No	0.0005	0.29	0.23
Standard worst-case ℓ_∞	No	0.001	0.30	0.24
Standard worst-case ℓ_∞	No	0.002	0.30	0.24
Standard worst-case ℓ_∞	No	0.004	0.29	0.23
Adaptive avg-case ℓ_∞	No	0.1	-0.36	-0.47
Adaptive avg-case ℓ_∞	No	0.2	**-0.53**	**-0.56**
Adaptive avg-case ℓ_∞	No	0.4	-0.41	-0.41
Adaptive avg-case ℓ_∞	No	0.8	-0.20	-0.18
Adaptive worst-case ℓ_∞	No	0.001	-0.36	-0.42
Adaptive worst-case ℓ_∞	No	0.002	-0.05	-0.10
Adaptive worst-case ℓ_∞	No	0.004	0.25	0.20
Adaptive worst-case ℓ_∞	No	0.008	0.26	0.24
Adaptive avg-case ℓ_∞	Yes	0.1	0.18	0.07
Adaptive avg-case ℓ_∞	Yes	0.2	0.05	-0.06
Adaptive avg-case ℓ_∞	Yes	0.4	-0.23	-0.37
Adaptive avg-case ℓ_∞	Yes	0.8	-0.46	**-0.62**
Adaptive worst-case ℓ_∞	Yes	0.001	0.30	0.18
Adaptive worst-case ℓ_∞	Yes	0.002	0.29	0.16
Adaptive worst-case ℓ_∞	Yes	0.004	0.21	0.07
Adaptive worst-case ℓ_∞	Yes	0.008	-0.04	-0.19

Table 7.4: A summary of correlation between sharpness and generalization for all experiments on **CIFAR-10** for ViTs trained from scratch. We boldface entries with $|\tau| > 0.5$ suggesting a reasonably strong correlation. LogitNorm stands for *logit normalization*.

Vision transformers trained from scratch on CIFAR-10

			Rank correlation coefficient τ	
Sharpness	**LogitNorm**	ρ	**CIFAR-10**	**CIFAR-10-C**
Standard avg-case ℓ_2	No	0.005	-0.45	**-0.54**
Standard avg-case ℓ_2	No	0.01	-0.39	-0.49
Standard avg-case ℓ_2	No	0.02	-0.20	-0.31
Standard avg-case ℓ_2	No	0.04	-0.08	-0.20
Standard worst-case ℓ_2	No	0.025	**-0.59**	**-0.62**
Standard worst-case ℓ_2	No	0.05	-0.37	-0.43
Standard worst-case ℓ_2	No	0.1	-0.16	-0.24
Standard worst-case ℓ_2	No	0.2	-0.12	-0.20
Adaptive avg-case ℓ_2	No	0.1	-0.45	**-0.50**
Adaptive avg-case ℓ_2	No	0.2	-0.45	-0.45
Adaptive avg-case ℓ_2	No	0.4	-0.42	-0.47
Adaptive avg-case ℓ_2	No	0.8	-0.10	0.08
Adaptive worst-case ℓ_2	No	0.5	**-0.64**	**-0.53**
Adaptive worst-case ℓ_2	No	1.0	-0.32	-0.19
Adaptive worst-case ℓ_2	No	2.0	-0.11	-0.01
Adaptive worst-case ℓ_2	No	4.0	-0.07	-0.03
Adaptive avg-case ℓ_2	Yes	0.1	-0.18	-0.31
Adaptive avg-case ℓ_2	Yes	0.2	-0.28	-0.40
Adaptive avg-case ℓ_2	Yes	0.4	-0.39	-0.46
Adaptive avg-case ℓ_2	Yes	0.8	-0.44	**-0.52**
Adaptive worst-case ℓ_2	Yes	0.25	-0.21	-0.12
Adaptive worst-case ℓ_2	Yes	0.5	-0.24	-0.17
Adaptive worst-case ℓ_2	Yes	1.0	-0.22	-0.19
Adaptive worst-case ℓ_2	Yes	2.0	-0.14	-0.11
Standard avg-case ℓ_∞	No	0.01	-0.44	**-0.54**
Standard avg-case ℓ_∞	No	0.02	-0.35	-0.45
Standard avg-case ℓ_∞	No	0.04	-0.17	-0.28
Standard avg-case ℓ_∞	No	0.08	-0.04	-0.14
Standard worst-case ℓ_∞	No	0.00001	**-0.61**	**-0.63**
Standard worst-case ℓ_∞	No	0.00002	-0.46	**-0.51**
Standard worst-case ℓ_∞	No	0.00004	-0.25	-0.31
Standard worst-case ℓ_∞	No	0.00008	-0.16	-0.22
Adaptive avg-case ℓ_∞	No	0.1	-0.45	**-0.53**
Adaptive avg-case ℓ_∞	No	0.2	-0.46	**-0.50**
Adaptive avg-case ℓ_∞	No	0.4	-0.45	-0.44
Adaptive avg-case ℓ_∞	No	0.8	-0.41	-0.47
Adaptive worst-case ℓ_∞	No	0.0005	**-0.68**	**-0.63**
Adaptive worst-case ℓ_∞	No	0.001	-0.43	-0.40
Adaptive worst-case ℓ_∞	No	0.002	-0.26	-0.23
Adaptive worst-case ℓ_∞	No	0.004	-0.18	-0.18
Adaptive avg-case ℓ_∞	Yes	0.1	-0.11	-0.23
Adaptive avg-case ℓ_∞	Yes	0.2	-0.16	-0.29
Adaptive avg-case ℓ_∞	Yes	0.4	-0.31	-0.42
Adaptive avg-case ℓ_∞	Yes	0.8	-0.40	-0.47
Adaptive worst-case ℓ_∞	Yes	0.0005	-0.20	-0.23
Adaptive worst-case ℓ_∞	Yes	0.001	-0.22	-0.26
Adaptive worst-case ℓ_∞	Yes	0.002	-0.29	-0.34
Adaptive worst-case ℓ_∞	Yes	0.004	-0.39	-0.44

7.8 Omitted Proofs

7.8.1 Asymptotic Analysis of Adaptive Sharpness Measures

For the convenience of the reader we repeat here quickly the definitions of adaptive sharpness measures. Let $L_{\mathcal{S}}(\boldsymbol{w}) = \frac{1}{|S|}\sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{S}} \ell_{\boldsymbol{xy}}(\boldsymbol{w})$ be the loss on a set of *training* points $\mathcal{S}$. For arbitrary weights $\boldsymbol{w}$ (i.e., not necessarily a minimum), then the *average-case* and *worst-case m-sharpness* is defined as:

$$\begin{aligned} S^{\rho}_{avg,p}(\boldsymbol{w},\boldsymbol{c}) &\triangleq \mathbb{E}_{\substack{\mathcal{S}\sim P_m \\ \boldsymbol{\delta}\sim\mathcal{N}(0,\rho^2 diag(\boldsymbol{c}^2))}} L_{\mathcal{S}}(\boldsymbol{w}+\boldsymbol{\delta}) - L_{\mathcal{S}}(\boldsymbol{w}) \\ S^{\rho}_{max,p}(\boldsymbol{w},\boldsymbol{c}) &\triangleq \mathbb{E}_{\mathcal{S}\sim P_m} \max_{\|\boldsymbol{\delta}\odot\boldsymbol{c}^{-1}\|_p\leq\rho} L_{\mathcal{S}}(\boldsymbol{w}+\boldsymbol{\delta}) - L_{\mathcal{S}}(\boldsymbol{w}), \end{aligned} \tag{7.7}$$

where $\odot/^{-1}$ denotes elementwise multiplication/inversion and P_m is the data distribution that returns m training pairs $(\boldsymbol{x},\boldsymbol{y})$.

If $\boldsymbol{c} = |\boldsymbol{w}|$ then the perturbation set is $\left\|\delta\odot|\boldsymbol{w}|^{-1}\right\|_p \leq \rho$. We first introduce a new variable $\boldsymbol{\gamma} = \boldsymbol{\delta}\odot|\boldsymbol{w}|^{-1}$ and do a Taylor expansion around w:

$$\begin{aligned} L_{\mathcal{S}}(\boldsymbol{w}+\boldsymbol{\delta}) = L_{\mathcal{S}}(\boldsymbol{w}+\boldsymbol{\gamma}\odot|\boldsymbol{w}|) = L_{\mathcal{S}}(\boldsymbol{w}) + \langle\nabla L_{\mathcal{S}}(\boldsymbol{w}), |\boldsymbol{w}|\odot\boldsymbol{\gamma}\rangle \\ + \frac{1}{2}\left\langle\boldsymbol{\gamma}\odot|\boldsymbol{w}|, \nabla^2 L_{\mathcal{S}}(\boldsymbol{w})\boldsymbol{\gamma}\odot|\boldsymbol{w}|\right\rangle + O(\|\boldsymbol{\gamma}\|_p^3), \end{aligned}$$

where $\nabla^2 L_{\mathcal{S}}(\boldsymbol{w})$ denotes the Hessian of $L_{\mathcal{S}}$ at $\boldsymbol{w}$.

Proposition 7.8.1. *Let $L_{\mathcal{S}} \in C^3(\mathbb{R}^s)$, S be a finite sample of training points $(x_i,y_i)_{i=1}^n$ and let P_m denote the uniform distribution over subsamples of size $m \leq n$ from S. Then we define for $p \geq 1$, $q \in \mathbb{R}$ such that $\frac{1}{p}+\frac{1}{q}=1$, then it holds*

$$\begin{aligned} &\lim_{\rho\to 0} S^{\rho}_{max,p}(\boldsymbol{w},|\boldsymbol{w}|) \\ &= \mathbb{E}_{\mathcal{S}\sim P_m}\begin{cases} \|\nabla L_{\mathcal{S}}(\boldsymbol{w})\odot|\boldsymbol{w}|\|_q\,\rho + O(\rho^2) & \text{if } \nabla L_{\mathcal{S}}(\boldsymbol{w})\odot|\boldsymbol{w}| \neq 0, \\ \frac{\rho^2}{2}\max_{\gamma\neq 0}\frac{\left\langle\gamma,\left(\nabla^2 L_{\mathcal{S}}(\boldsymbol{w})\odot(|\boldsymbol{w}||\boldsymbol{w}|^T)\right)\gamma\right\rangle}{\|\gamma\|_p^2} + O(\rho^3) & \text{if } \nabla L_{\mathcal{S}}(\boldsymbol{w})\odot|\boldsymbol{w}| = 0 \text{ and} \\ & \quad \nabla^2 L_{\mathcal{S}}(\boldsymbol{w})\odot(|\boldsymbol{w}||\boldsymbol{w}|^T) \text{ not negative definite} \\ O(\rho^3) & \text{if } \nabla L_{\mathcal{S}}(\boldsymbol{w})\odot|\boldsymbol{w}| = 0 \text{ and} \\ & \quad \nabla^2 L_{\mathcal{S}}(\boldsymbol{w})\odot(|\boldsymbol{w}||\boldsymbol{w}|^T) \text{ is negative definite} \end{cases} \end{aligned}$$

Proof. We get

$$\max_{\|\gamma\|_p\leq\rho} L_{\mathcal{S}}(\boldsymbol{w}+\boldsymbol{\gamma}\odot|\boldsymbol{w}|) - L_{\mathcal{S}}(\boldsymbol{w})$$

$$= \max_{\|\gamma\|_p \leq \rho} \langle \nabla L_{\mathcal{S}}(\boldsymbol{w}), |\boldsymbol{w}| \odot \boldsymbol{\gamma} \rangle + \frac{1}{2} \left\langle \boldsymbol{\gamma} \odot |\boldsymbol{w}|, \nabla^2 L_{\mathcal{S}}(\boldsymbol{w}) \boldsymbol{\gamma} \odot |\boldsymbol{w}| \right\rangle + O(\|\boldsymbol{\gamma}\|_p^3)$$
$$= \max_{\|\gamma\|_p \leq \rho} \langle \nabla L_{\mathcal{S}}(\boldsymbol{w}) \odot |\boldsymbol{w}|, \boldsymbol{\gamma} \rangle + \frac{1}{2} \left\langle \boldsymbol{\gamma}, \left(\nabla^2 L_{\mathcal{S}}(\boldsymbol{w}) \odot (|\boldsymbol{w}||\boldsymbol{w}|^T) \right) \boldsymbol{\gamma} \right\rangle + O(\|\boldsymbol{\gamma}\|_p^3)$$

If $\nabla L_{\mathcal{S}}(\boldsymbol{w}) \odot |\boldsymbol{w}| \neq 0$, then the first order term dominates for ρ sufficiently small and we get

$$\max_{\|\gamma\|_p \leq \rho} \langle \nabla L_{\mathcal{S}}(\boldsymbol{w}) \odot |\boldsymbol{w}|, \boldsymbol{\gamma} \rangle = \max_{\|\gamma\|_p \leq \rho} \| \nabla L_{\mathcal{S}}(\boldsymbol{w}) \odot |\boldsymbol{w}| \|_q \, \|\boldsymbol{\gamma}\|_p = \rho \, \| \nabla L_{\mathcal{S}}(\boldsymbol{w}) \odot |\boldsymbol{w}| \|_q \, .$$

Otherwise we have to consider

$$\max_{\|\gamma\|_p \leq \rho} \frac{1}{2} \left\langle \boldsymbol{\gamma}, \left(\nabla^2 L_{\mathcal{S}}(\boldsymbol{w}) \odot (|\boldsymbol{w}||\boldsymbol{w}|^T) \right) \boldsymbol{\gamma} \right\rangle .$$

If $\nabla^2 L(\boldsymbol{w}) \odot (|\boldsymbol{w}||\boldsymbol{w}|^T)$ is negative definite, then the maximum is zero attained at $\boldsymbol{\gamma} = 0$. In the other case, we get

$$\max_{\|\gamma\|_p \leq \rho} \frac{1}{2} \left\langle \boldsymbol{\gamma}, \left(\nabla^2 L_{\mathcal{S}}(\boldsymbol{w}) \odot (|\boldsymbol{w}||\boldsymbol{w}|^T) \right) \boldsymbol{\gamma} \right\rangle = \frac{\rho^2}{2} \max_{\gamma \neq 0} \frac{\left\langle \boldsymbol{\gamma}, \left(\nabla^2 L_{\mathcal{S}}(\boldsymbol{w}) \odot (|\boldsymbol{w}||\boldsymbol{w}|^T) \right) \boldsymbol{\gamma} \right\rangle}{\|\boldsymbol{\gamma}\|_p^2}.$$

This almost finishes the proof. Finally, it holds

$$\lim_{\rho \to 0} S^{\rho}_{max,p}(\boldsymbol{w}, |\boldsymbol{w}|) = \lim_{\rho \to 0} \mathbb{E}_{\mathcal{S} \sim P_m} \left[\max_{\|\gamma\|_p \leq \rho} L_{\mathcal{S}}(\boldsymbol{w} + \boldsymbol{\gamma} \odot |\boldsymbol{w}|) - L_{\mathcal{S}}(\boldsymbol{w}) \right],$$
$$= \mathbb{E}_{\mathcal{S} \sim P_m} \left[\lim_{\rho \to 0} \max_{\|\gamma\|_p \leq \rho} L_{\mathcal{S}}(\boldsymbol{w} + \boldsymbol{\gamma} \odot |\boldsymbol{w}|) - L_{\mathcal{S}}(\boldsymbol{w}) \right]$$

where for the last step we have used that $\mathbb{E}_{\mathcal{S} \sim P_m}$ is the expectation over all possible subsamples of size m and thus boils down to a finite sum for which we can drag the limit inside. □

We note that for $p = 2$ it holds $q = 2$ and

$$\max_{\gamma \neq 0} \frac{\left\langle \boldsymbol{\gamma}, \left(\nabla^2 L_{\mathcal{S}}(\boldsymbol{w}) \odot (|\boldsymbol{w}||\boldsymbol{w}|^T) \right) \boldsymbol{\gamma} \right\rangle}{\|\boldsymbol{\gamma}\|_2^2} = \lambda_{\max} \Big(\nabla^2 L_{\mathcal{S}}(\boldsymbol{w}) \odot (|\boldsymbol{w}||\boldsymbol{w}|^T) \Big),$$

which is the result used in the main paper.

Proposition 7.8.2. *Let $L_{\mathcal{S}} \in C^3(\mathbb{R}^s)$, S be a finite sample of training points $(x_i, y_i)_{i=1}^n$ and let P_m denote the uniform distribution over subsamples of size $m \leq n$ from S. Then*

$$\lim_{\rho \to 0} \frac{2}{\rho^2} S^{\rho}_{avg}(\boldsymbol{w}, |\boldsymbol{w}|) = \mathbb{E}_{\mathcal{S} \sim P_m} \left[tr(\nabla^2 L_{\mathcal{S}}(\boldsymbol{w}) \odot |\boldsymbol{w}||\boldsymbol{w}|^{\top}) \right] + O(\rho)$$

Proof. Let us consider the loss without the subcript for clarity. Then we consider

$$\mathbb{E}_{\boldsymbol{\delta}\sim\mathcal{N}(0,\rho^2 diag(c^2))}\, L_{\mathcal{S}}(\boldsymbol{w}+\boldsymbol{\delta}) - L_{\mathcal{S}}(\boldsymbol{w})$$

When plugging in the Taylor expansion of the loss, we see that

$$\begin{aligned}
&\mathbb{E}_{\boldsymbol{\delta}\sim\mathcal{N}(0,\rho^2 diag(c^2))} L_{\mathcal{S}}(\boldsymbol{w}+\boldsymbol{\delta}) - L_{\mathcal{S}}(\boldsymbol{w}) \\
&\quad = \mathbb{E}_{\boldsymbol{\gamma}\in\mathcal{N}(0,\rho^2\boldsymbol{I})}\left[\langle \nabla L_{\mathcal{S}}(\boldsymbol{w}), |\boldsymbol{w}|\odot\boldsymbol{\gamma}\rangle + \frac{1}{2}\left\langle \boldsymbol{\gamma}\odot|\boldsymbol{w}|, \nabla^2 L_{\mathcal{S}}(\boldsymbol{w})\boldsymbol{\gamma}\odot|\boldsymbol{w}|\right\rangle + O(\|\boldsymbol{\gamma}\|_2^3)\right] \\
&\quad = \frac{1}{2}\,\mathbb{E}_{\boldsymbol{\gamma}\in\mathcal{N}(0,\rho^2\boldsymbol{I})}\left[\left\langle \boldsymbol{\gamma}\odot|\boldsymbol{w}|, \nabla^2 L_{\mathcal{S}}(\boldsymbol{w})\boldsymbol{\gamma}\odot|\boldsymbol{w}|\right\rangle\right] + O(\rho^3) \\
&\quad = \frac{1}{2}\,\mathbb{E}_{\boldsymbol{\gamma}\in\mathcal{N}(0,\rho^2\boldsymbol{I})}\left[\left\langle \boldsymbol{\gamma}, \left(\nabla^2 L_{\mathcal{S}}(\boldsymbol{w})\odot|\boldsymbol{w}||\boldsymbol{w}|^T\right)\boldsymbol{\gamma}\right\rangle\right] + O(\rho^3) \\
&\quad = \frac{\rho^2}{2}\mathrm{tr}(\nabla^2 L_{\mathcal{S}}(\boldsymbol{w})\odot|\boldsymbol{w}||\boldsymbol{w}|^\top) + O(\rho^3)
\end{aligned}$$

where we use that the components of $\boldsymbol{\gamma}$ are independent and have zero mean and thus the first order term vanishes and for the second order term only the diagonal entries remain which are equal to the variance ρ^2. Finally, we take the expectation with respect to P_m. As in the proof of Proposition 7.8.1 we can drag the limit inside as the expectation with respect to P_m corresponds to a finite sum. □

7.8.2 Derivations for Diagonal Linear Networks

Hessian for diagonal linear networks. Denote $\boldsymbol{r} = \boldsymbol{X}(\boldsymbol{u}\odot\boldsymbol{v}) - \boldsymbol{y}$, $\boldsymbol{V} = \mathrm{diag}(\boldsymbol{v})$, $\boldsymbol{U} = \mathrm{diag}(\boldsymbol{u})$, then the Hessian of the loss $\nabla^2 L(\boldsymbol{w})$ for diagonal linear networks is given by:

$$L(\boldsymbol{w}) = \begin{bmatrix} \boldsymbol{V}\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{V} & \boldsymbol{V}\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{U} + \mathrm{diag}(\boldsymbol{X}^\top\boldsymbol{r}) \\ \boldsymbol{V}\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{U} + \mathrm{diag}(\boldsymbol{X}^\top\boldsymbol{r}) & \boldsymbol{U}\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{U} \end{bmatrix}. \tag{7.8}$$

It is easy to verify that the data-dependent terms disappear due to the assumption of whitened data $\boldsymbol{X}^\top\boldsymbol{X} = \boldsymbol{I}$ and zero residuals $\boldsymbol{r}$ at a minimum. Thus, we arrive at a much simpler expression for the Hessian:

$$L(\boldsymbol{w}) = \begin{bmatrix} \mathrm{diag}(\boldsymbol{v}\odot\boldsymbol{v}) & \mathrm{diag}(\boldsymbol{v}\odot\boldsymbol{u}) \\ \mathrm{diag}(\boldsymbol{v}\odot\boldsymbol{u}) & \mathrm{diag}(\boldsymbol{u}\odot\boldsymbol{u}) \end{bmatrix}, \tag{7.9}$$

Maximum eigenvalue for diagonal linear networks. Since the Hessian has a simple block structure, we can rearrange the rows and columns coherently and get a block-

diagonal structure as follows

$$\begin{bmatrix} v_1^2 & v_1u_1 & 0 & \dots & 0 \\ v_1u_1 & u_1^2 & 0 & \dots & 0 \\ 0 & 0 & \dots & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & 0 & v_d^2 & v_du_d \\ 0 & \dots & 0 & v_du_d & u_d^2 \end{bmatrix} \tag{7.10}$$

where eigenvalues of each 2×2 submatrix are $u_i^2 + v_i^2$ and 0. Thus, $\lambda_{\max} = \max_{1 \leq i \leq d} v_i^2 + u_i^2$ by using the property of block-diagonal matrices.

7.9 Correlation Between Sharpness and Generalization Gap

Throughout the paper we focused on correlation between sharpness and *test error*, but it is natural to ask if the picture differs if we consider correlation between sharpness and *generalization gap*, i.e., the difference between the test error and training error. We note that in the experiments on CIFAR-10 in Section 7.6.1, since we consider only models with $\leq 1\%$ training error and since the test error is significantly larger than 1%, the behavior of generalization gap vs. sharpness has to be almost identical to that of test error vs. sharpness. For other datasets, however, the training error is not necessarily close to 0, thus in Figure 7.8 and Figure 7.9, we additionally plot the *generalization gap* vs. sharpness (and side-by-side the test error vs. sharpness for the sake of convenience) for the ImageNet experiments. We observe only small differences in the correlation values which do not alter the conclusions about the relationship of sharpness and generalization.

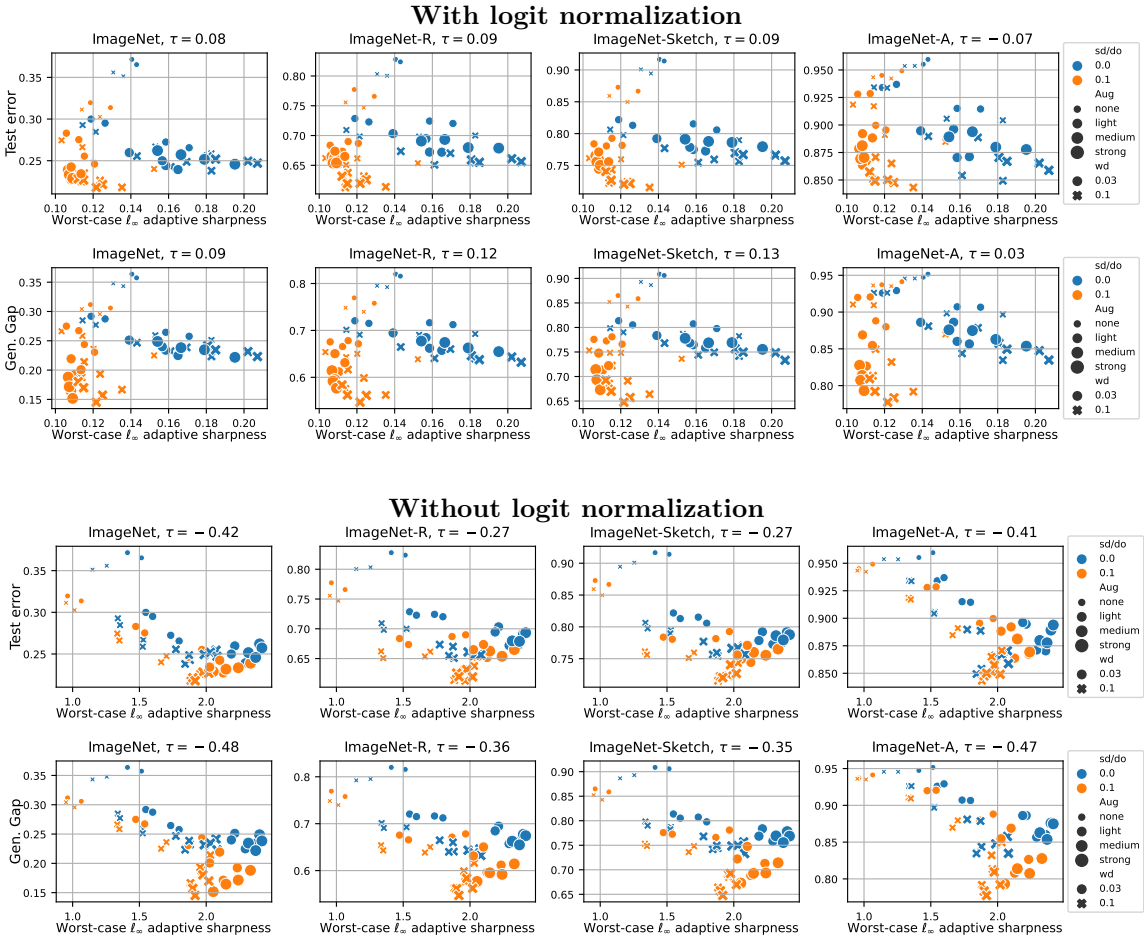


Figure 7.8: ViT-B/16 trained from scratch on ImageNet-1k. We show side-by-side the test error and **generalization gap** (**Gen. Gap**) for 56 models from Steiner et al. (2021) on ImageNet and its OOD variants vs. worst-case ℓ_∞ sharpness with (top) or without (bottom) normalization at $\rho = 0.002$. The color indicates models trained with stochastic depth (sd) and dropout (do), markers and their size indicate the strength of weight decay (wd) and augmentations (aug), and τ indicates the rank correlation coefficient.

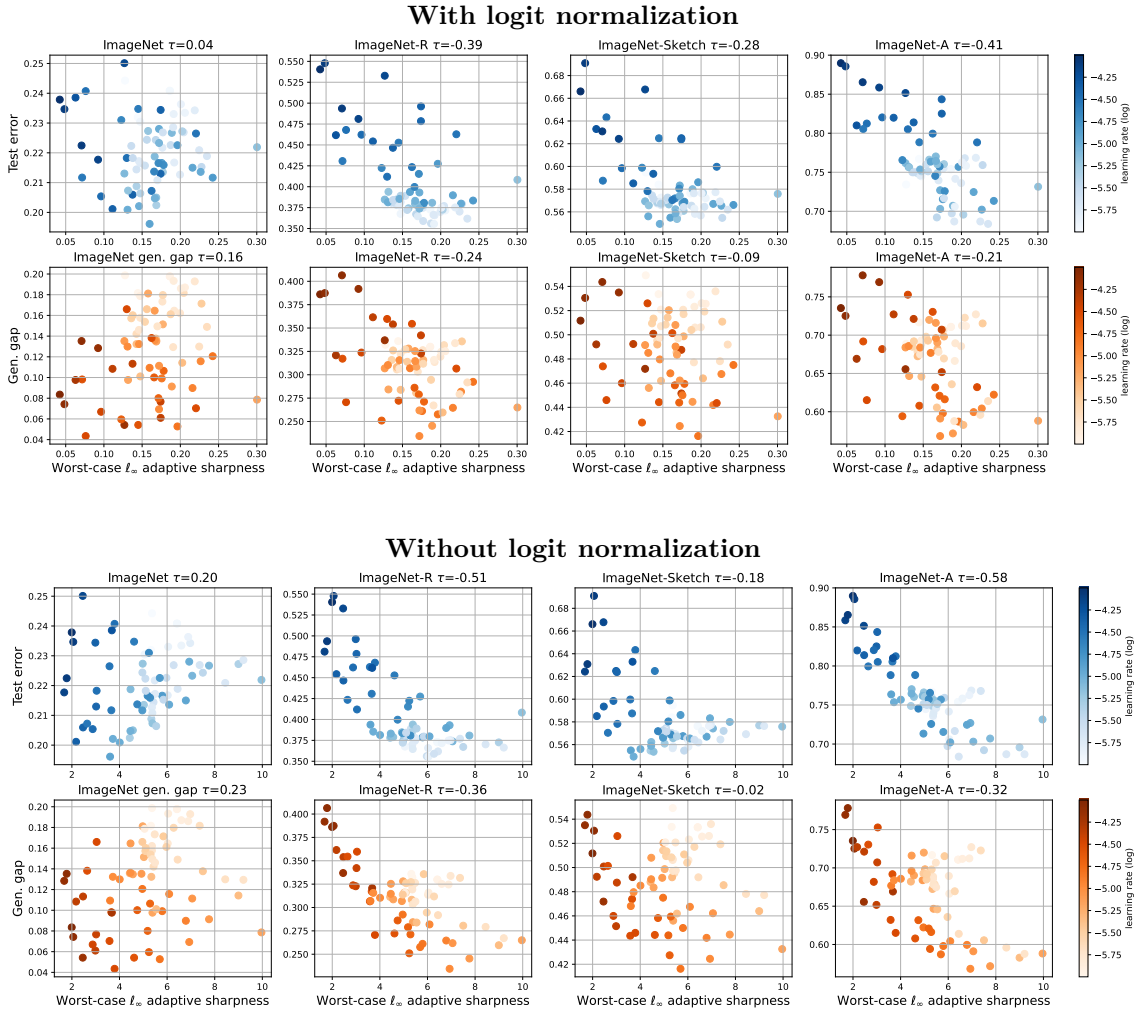


Figure 7.9: Fine-tuning CLIP ViT-B/32 on ImageNet-1k. We show side-by-side the test error and **generalization gap** (**Gen. gap**) for 72 models from Wortsman et al. (2022) on ImageNet and its OOD variants vs. worst-case ℓ_∞ sharpness with (top) or without (bottom) normalization at $\rho = 0.002$. Darker color indicates larger learning rate used for fine-tuning.

7.10 ImageNet-1k Models Trained from Scratch from Steiner et al. (2021): Extra Details and Figures

Experimental details. As explained in the main paper, the ViT-B/16-224 weights were trained on ImageNet-1k for 300 epochs with different hyperparameter settings, and subsequently fine-tuned on the same dataset for 20.000 steps with 2 different learning rates (0.01 and 0.03). The pretraining hyperparameters include 7 augmentation types (*none, light0, light1, medium0, medium1, strong0, strong1*), which we group into (*none, light, medium, strong*) in the plots. Weight decay was either 0.1 or 0.03, and dropout and stochastic depth were either both set to 0 or both set to 0.1. We evaluated the resulting 56 configurations. The model weights can be obtained from https://github.com/google-research/vision_transformer.

Sharpness evaluation. For sharpness evaluation we use 2048 data points from the training set split in 8 batches: we compute sharpness on each of them and report the average. For worst-case sharpness we use Auto-PGD for 20 steps (for each batch) with random uniform initialization in the feasible set, while for average-case sharpness we sample 100 different weights perturbations for every batch. We use the same sharpness evaluation for all ImageNet-1k and MNLI models. For convenience we restate the algorithm of Auto-PGD in Algorithm 5: it follows the original version presented in Croce and Hein (2020b) while using the network weights $\boldsymbol{w}$ as optimization variables instead of the input image components. In Alg. 5 we denote f the target objective function (cross-entropy loss on the batch of images in our experiments), S the feasible set of perturbations and P_S the projection onto it. Also, η and W are fixed hyperparameters (we keep the original values), and the two conditions in Line 13 can be found in Croce and Hein (2020b).

Algorithm 5: Auto-PGD

1: **Input:** objective function f, perturbation set S, $\boldsymbol{w}^{(0)}$, η, N_{iter}, $W = \{w_0, \dots, w_n\}$
2: **Output:** $\boldsymbol{w}_{\max}$, $f_{\max}$
3: $\boldsymbol{w}^{(1)} \leftarrow P_{\mathcal{S}}\left(\boldsymbol{w}^{(0)} + \eta \nabla f(\boldsymbol{w}^{(0)})\right)$
4: $f_{\max} \leftarrow \max\{f(\boldsymbol{w}^{(0)}), f(\boldsymbol{w}^{(1)})\}$
5: $\boldsymbol{w}_{\max} \leftarrow \boldsymbol{w}^{(0)}$ **if** $f_{\max} \equiv f(\boldsymbol{w}^{(0)})$ **else** $\boldsymbol{w}_{\max} \leftarrow \boldsymbol{w}^{(1)}$
6: **for** $k = 1$ **to** $N_{\text{iter}} - 1$ **do**
7: $\quad \boldsymbol{z}^{(k+1)} \leftarrow P_{\mathcal{S}}\left(\boldsymbol{w}^{(k)} + \eta \nabla f(\boldsymbol{w}^{(k)})\right)$
8: $\quad \boldsymbol{w}^{(k+1)} \leftarrow P_{\mathcal{S}}\left(\boldsymbol{w}^{(k)} + \alpha(\boldsymbol{z}^{(k+1)} - \boldsymbol{w}^{(k)}) + (1-\alpha)(\boldsymbol{w}^{(k)} - \boldsymbol{w}^{(k-1)})\right)$
9: $\quad$ **if** $f(\boldsymbol{w}^{(k+1)}) > f_{\max}$ **then**
10: $\quad\quad \boldsymbol{w}_{\max} \leftarrow \boldsymbol{w}^{(k+1)}$ and $f_{\max} \leftarrow f(\boldsymbol{w}^{(k+1)})$
11: $\quad$ **end if**
12: $\quad$ **if** $k \in W$ **then**
13: $\quad\quad$ **if** Condition 1 **or** Condition 2 **then**
14: $\quad\quad\quad \eta \leftarrow \eta/2$ and $\boldsymbol{w}^{(k+1)} \leftarrow \boldsymbol{w}_{\max}$
15: $\quad\quad$ **end if**
16: $\quad$ **end if**
17: **end for**

Extra figures. For each sharpness definition we show for three values of ρ the correlation between test error on ImageNet (in-distribution) and on the various distribution shifts. In particular, we use worst-case ℓ_∞ adaptive sharpness with (Fig. 7.10) and without (Fig. 7.11) logit normalization, and average-case adaptive sharpness with (Fig. 7.12) and without (Fig. 7.13) logit normalization. For all figures the color shows stochastic depth / dropout, the marker size corresponds to augmentation strength, and the marker type to weight decay. In addition to the OOD-datasets from the main paper, we here report the results for ImageNet-V2 (Recht et al., 2019) and ObjectNet (Barbu et al., 2019). ImageNet-V2 consists in a new test set for ImageNet models and is sampled from the same image distribution as the existing validation set: then, the performance of the classifiers on it are highly correlated to that on ImageNet validation set, and ImageNet-V2 cannot be considered a distribution shift in the same sense as the other datasets. In general, we observe that sharpness variants are not predictive of the performance on ImageNet and the OOD datasets, typically only separating models by stochastic depth / dropout, but not ranking them according to generalization properties, and often even yielding a negative correlation with OOD test error. The only case where low sharpness indicates low test-error is for logit-normalized average-case adaptive sharpness on ImageNet and ImageNet-v2. For the remaining OOD datasets, however, there are always models with low sharpness and larger test error.

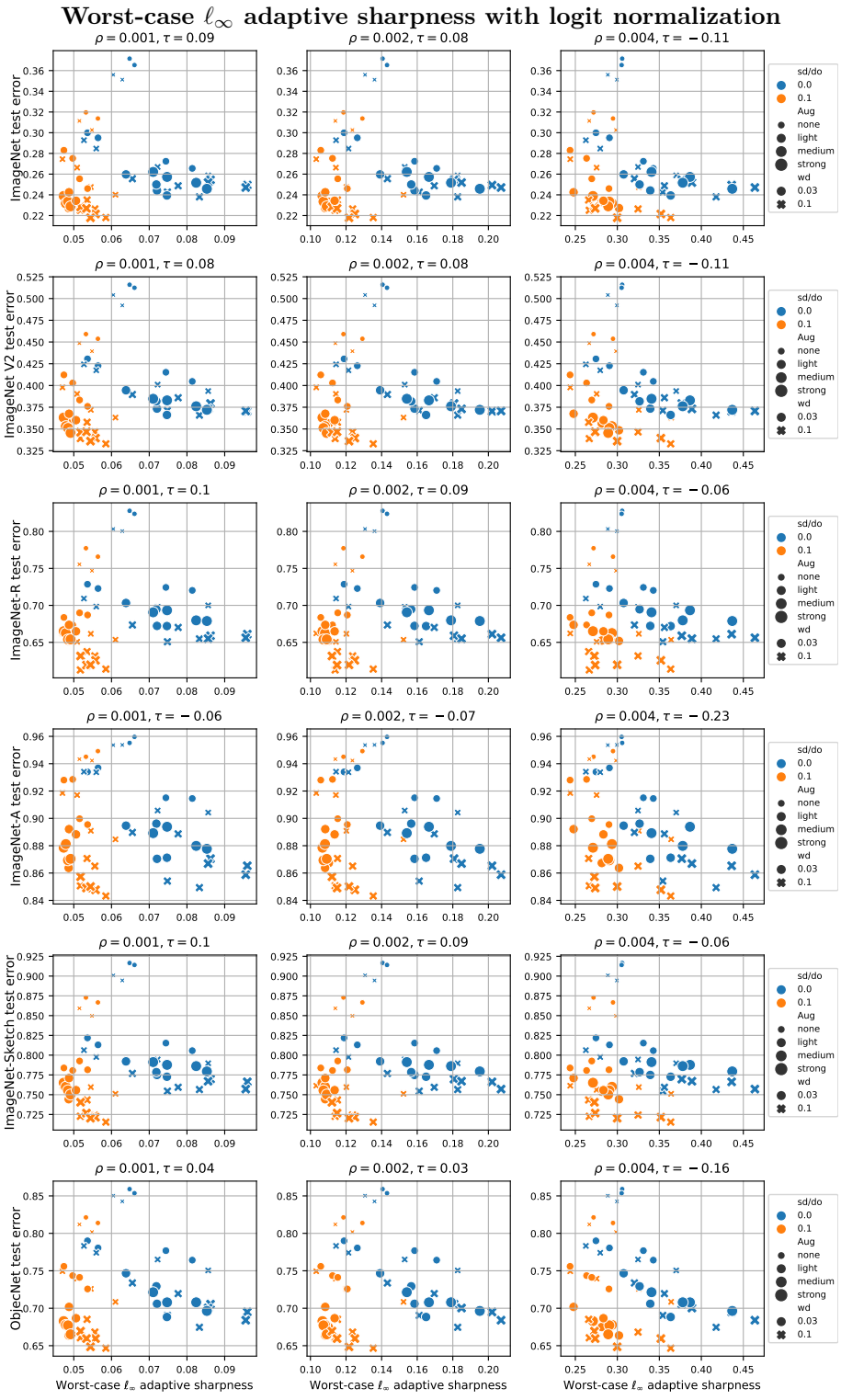


Figure 7.10: Correlation of sharpness with generalization on ImageNet for different ρ and for different distribution shifts.

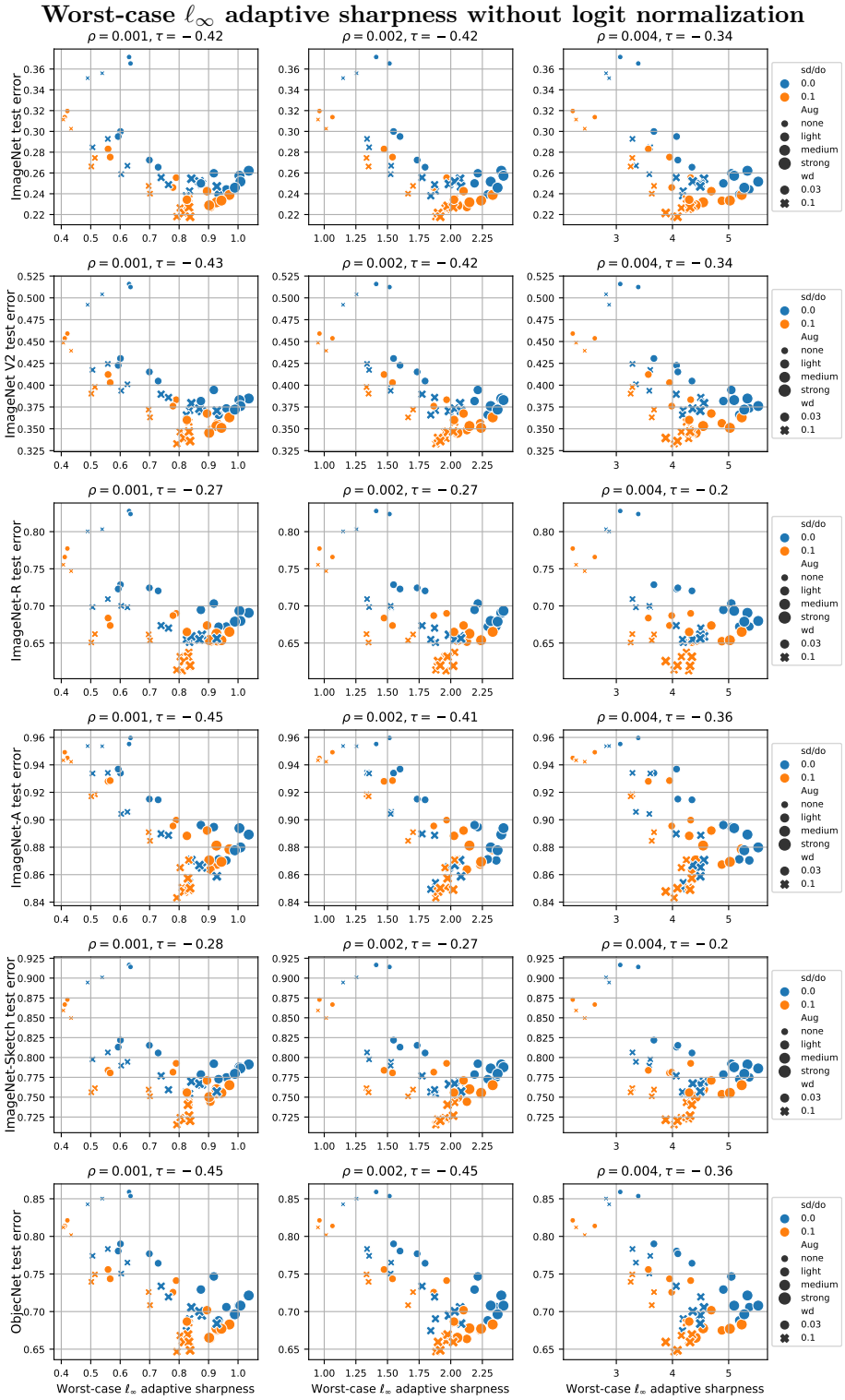


Figure 7.11: Correlation of sharpness with generalization on ImageNet for different ρ and for different distribution shifts.

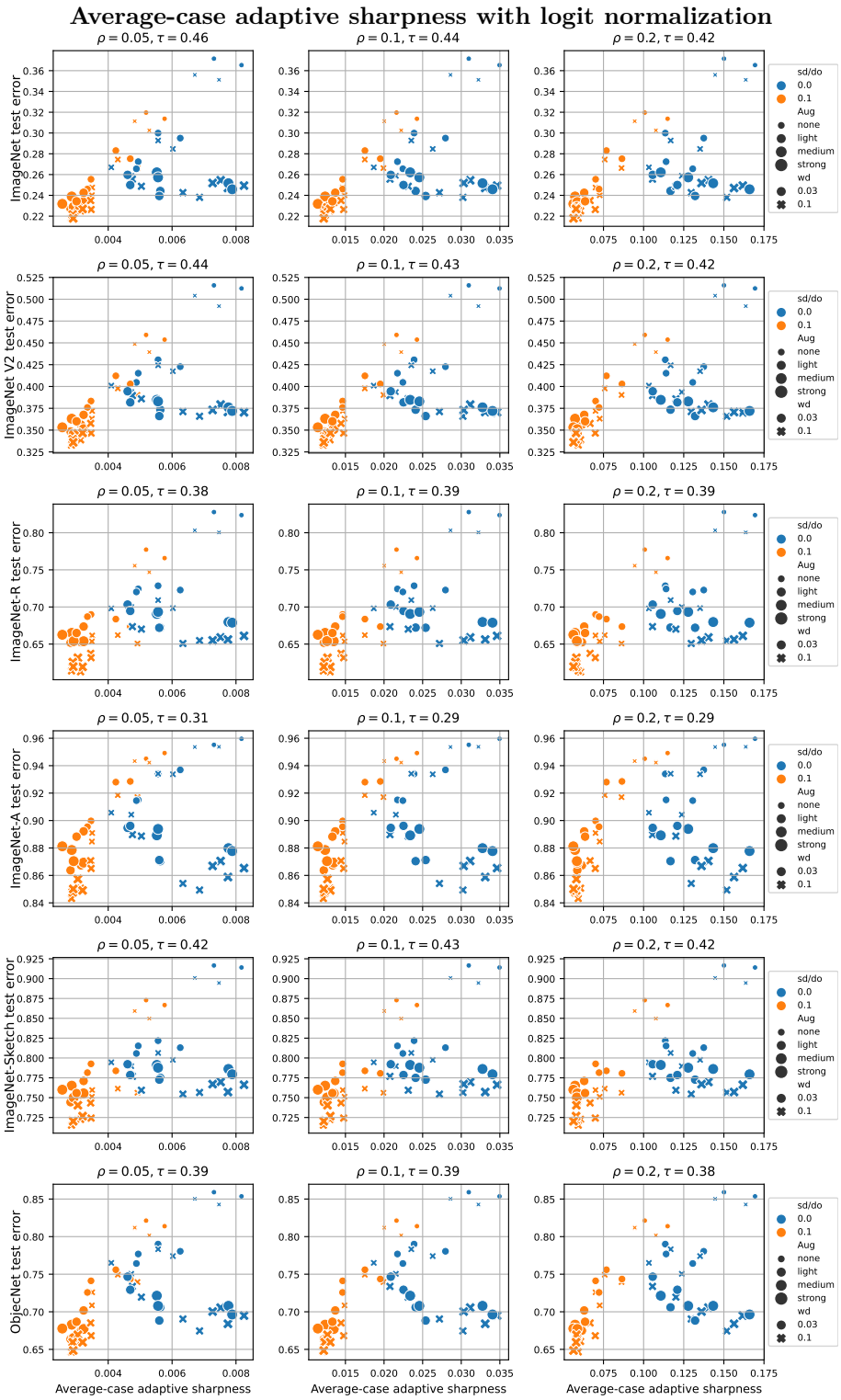


Figure 7.12: Correlation of sharpness with generalization on ImageNet for different ρ and for different distribution shifts.

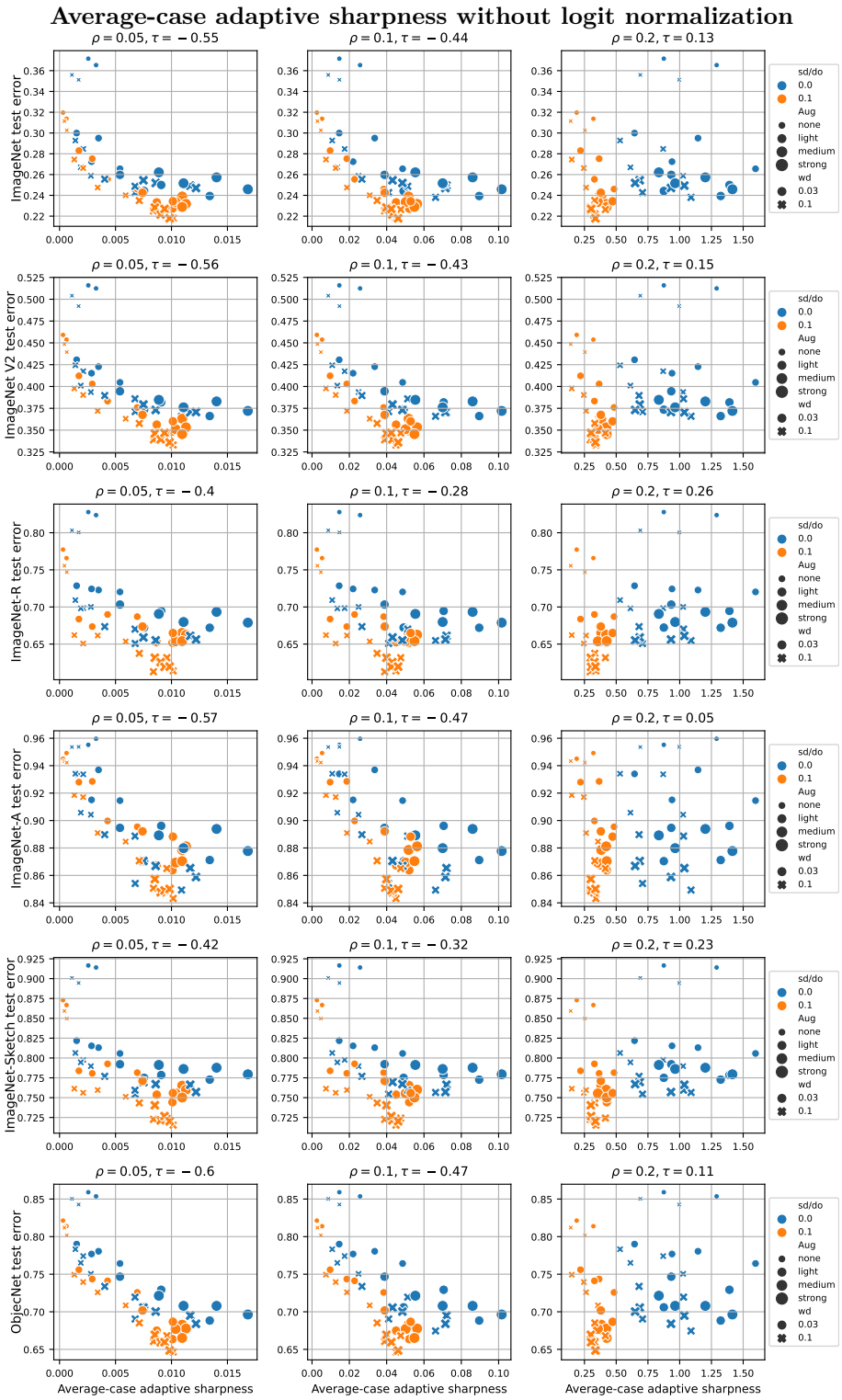


Figure 7.13: Correlation of sharpness with generalization on ImageNet for different ρ and for different distribution shifts.

7.11 Fine-tuning of ImageNet-1k Models Pretrained on ImageNet-21k from Steiner et al. (2021): Extra Figures and Details

Experimental details. All hyperparameter settings are identical to those explained in Appendix 7.10, only the pretraining dataset is ImageNet-21k instead of ImageNet-1k. Since two of the models showed close to 100% test error, we did not evaluate them, resulting in 54 instead of 56 models.

Extra figures. Like in Appendix 7.10 we show each sharpness definition for three values of ρ and its the correlation to test error on ImageNet (in-distribution) and on the various distribution shifts. The observations are very similar to those on ImageNet-1k pretraining: sharpness variants are not predictive of the performance on ImageNet and the distribution shift datasets, typically only separating models by stochastic depth / dropout, and often even yielding a negative correlation with OOD test error.

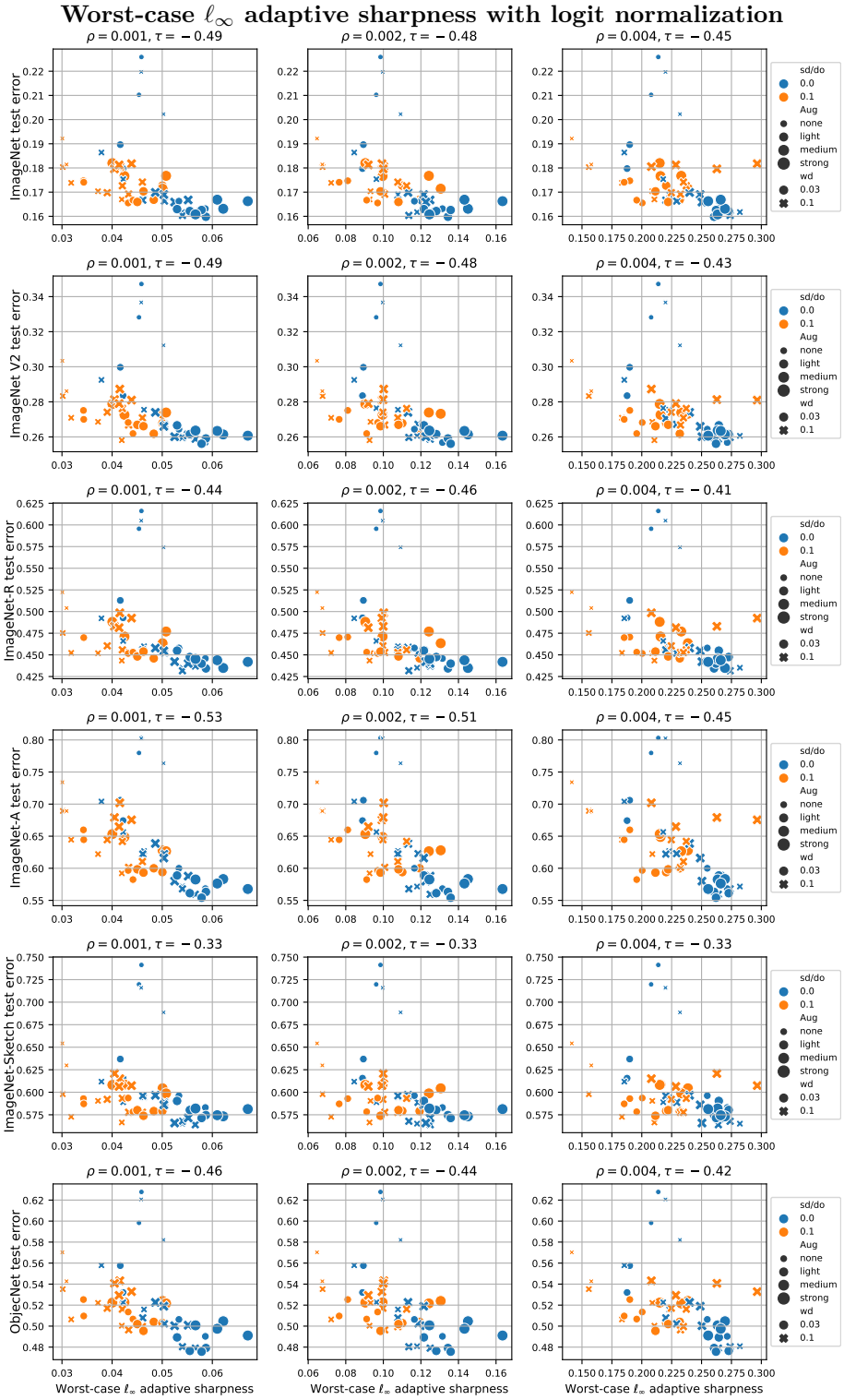


Figure 7.14: Correlation of sharpness with generalization on ImageNet for different ρ and for different distribution shifts.

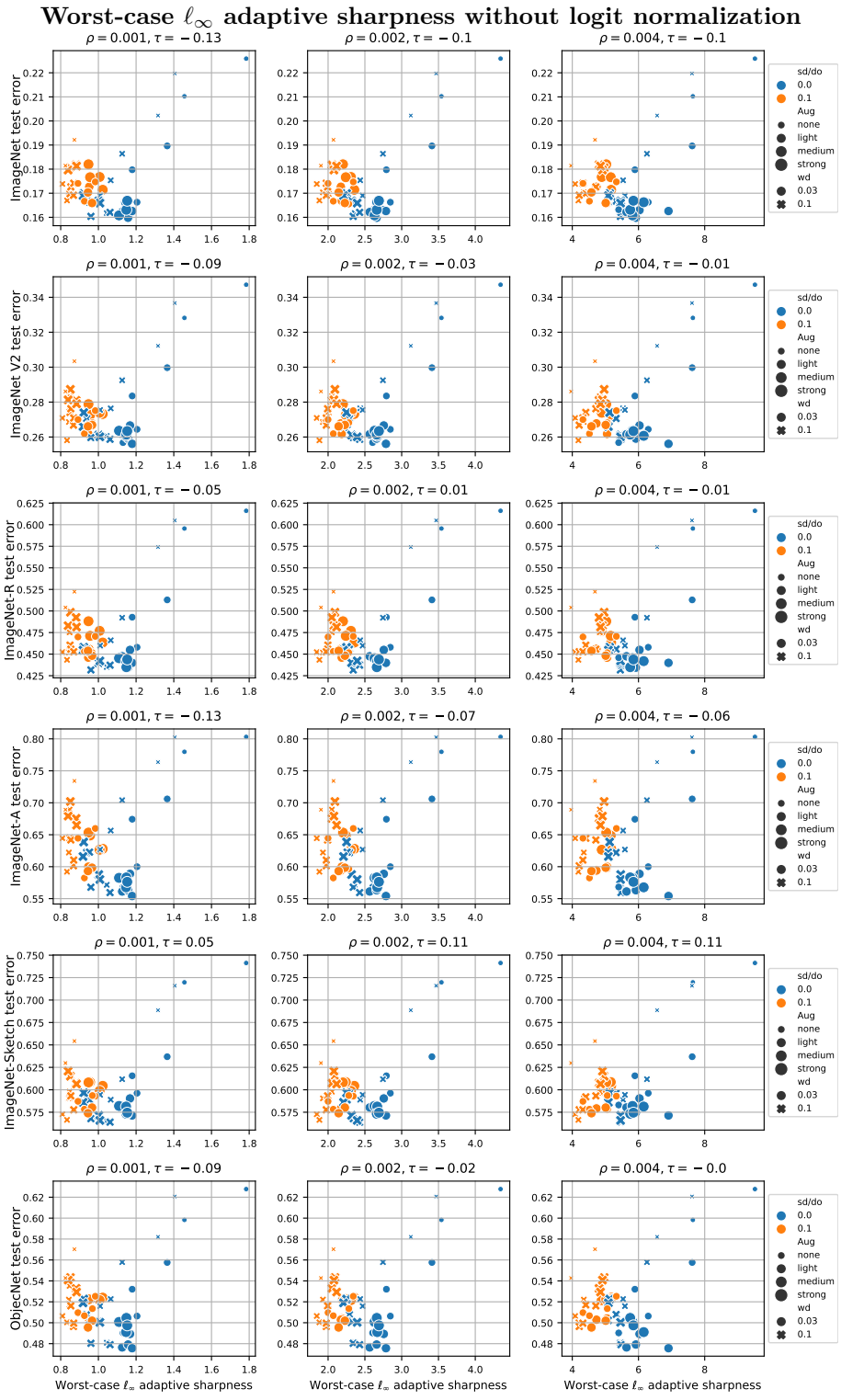


Figure 7.15: Correlation of sharpness with generalization on ImageNet for different ρ and for different distribution shifts.

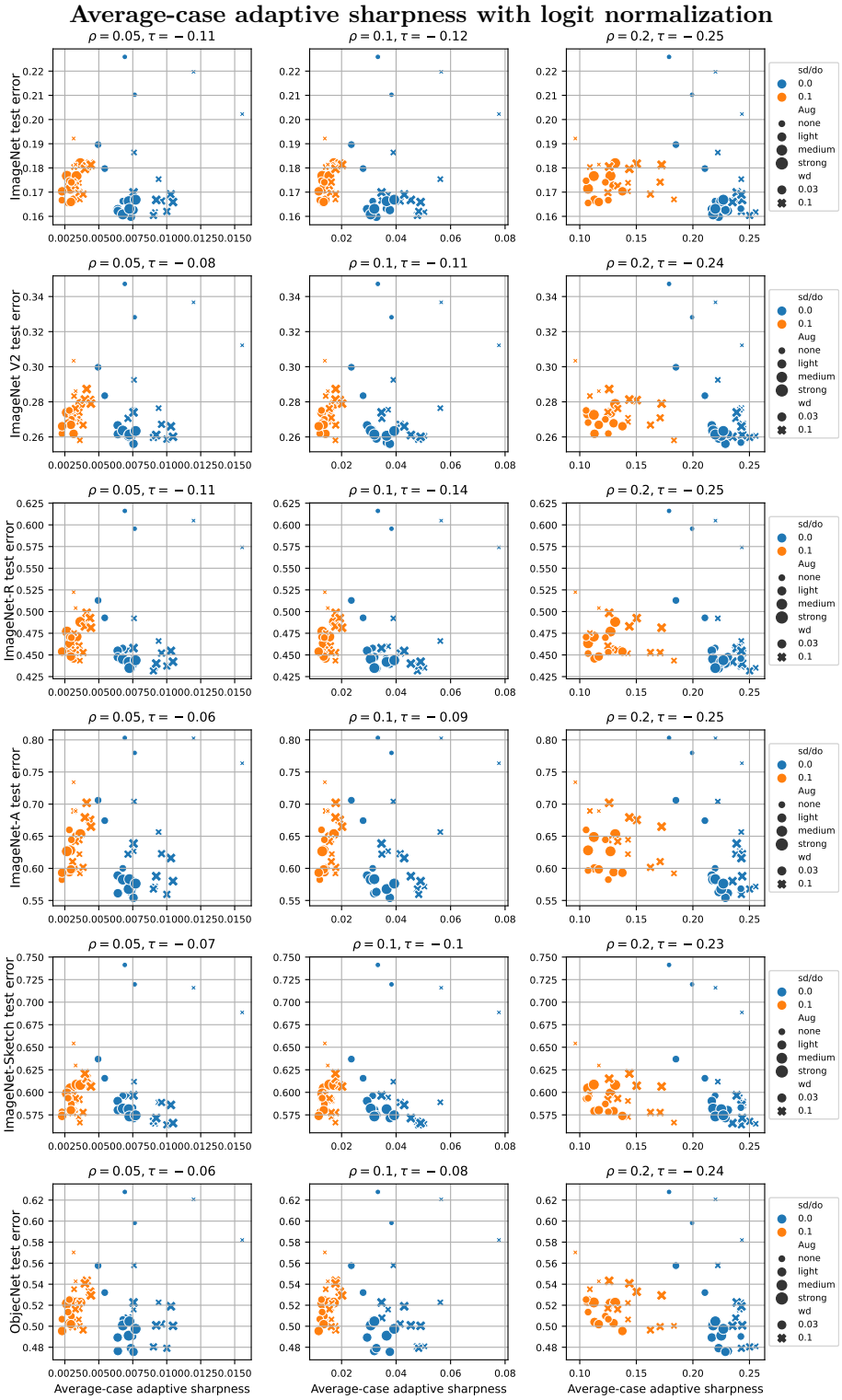


Figure 7.16: Correlation of sharpness with generalization on ImageNet for different ρ and for different distribution shifts.

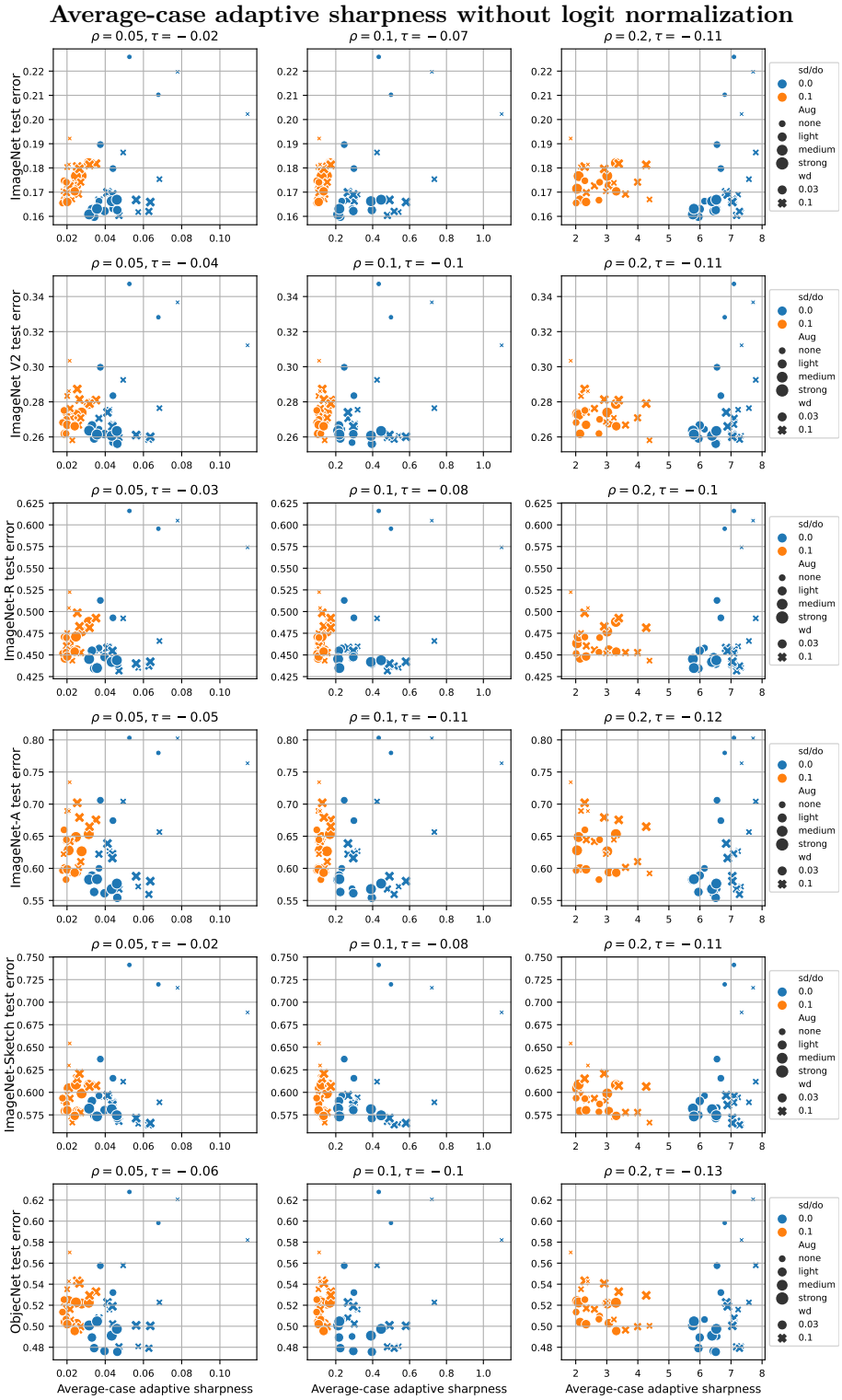


Figure 7.17: Correlation of sharpness with generalization on ImageNet for different ρ and for different distribution shifts.

7.12 ImageNet Models both Pretrained on ImageNet-1k and ImageNet-21k from Steiner et al. (2021)

For completeness, we here show for two sharpness definitions the models pretrained on ImageNet-21k and ImageNet-1k together. We find the better-generalizing models pretrained on ImageNet-21k to have significantly higher worst-case sharpness, and roughly equal or higher logit-normalized average-case adaptive sharpness, underlining that the models generalization properties resulting from different pretraining datasets are not captured.

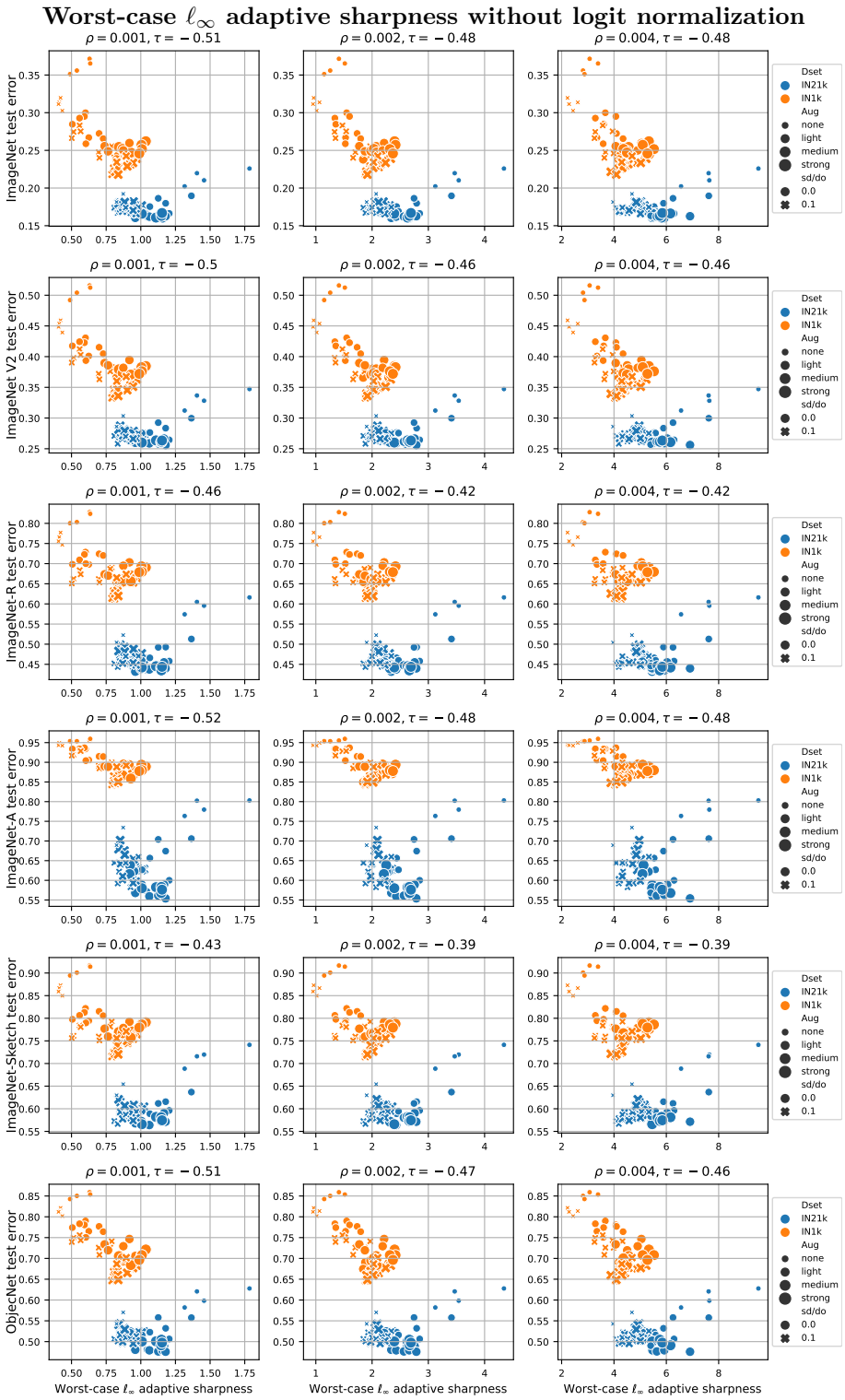


Figure 7.18: Correlation of sharpness with generalization on ImageNet-1k for different ρ and for different distribution shifts.

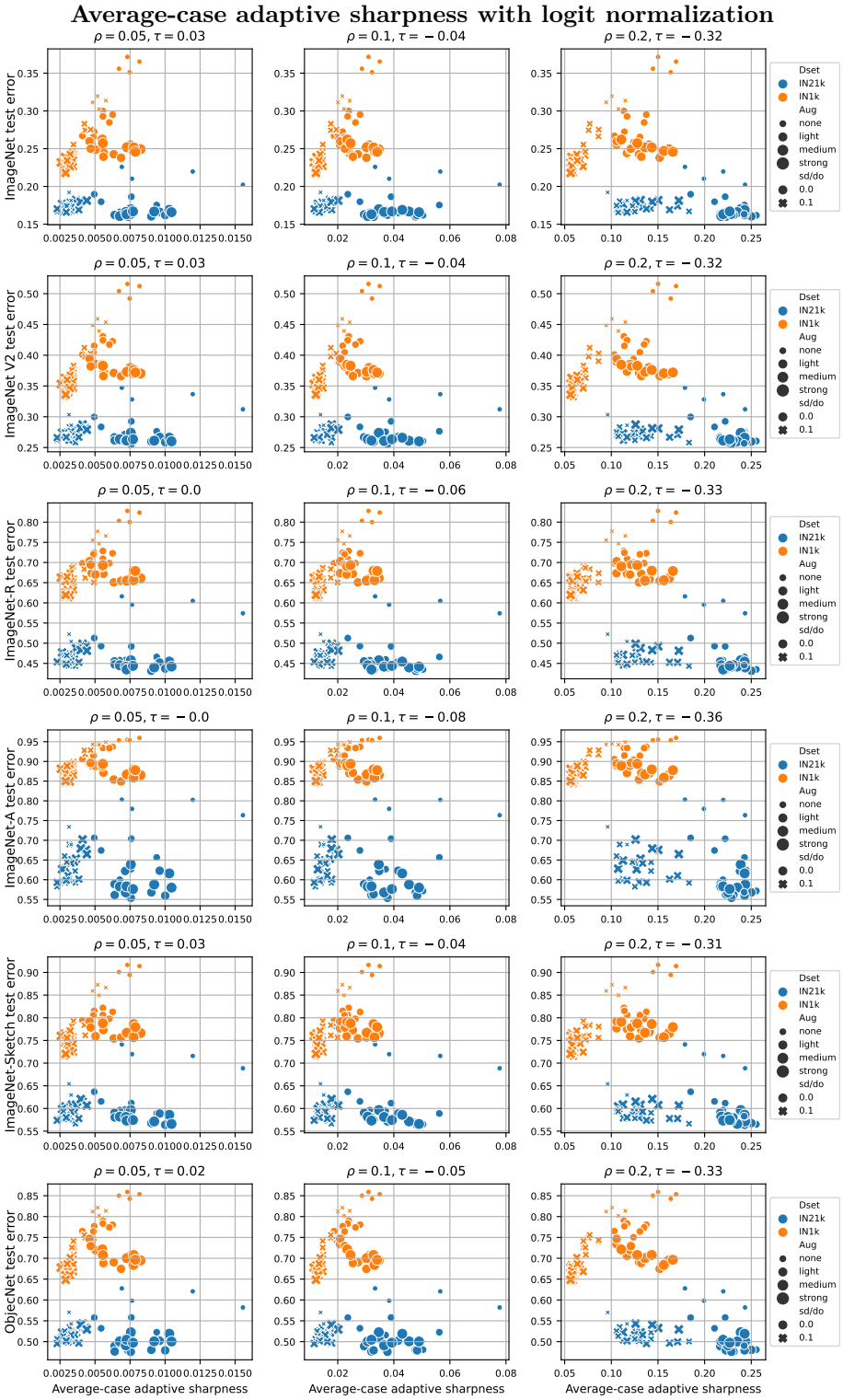


Figure 7.19: Correlation of sharpness with generalization on ImageNet-1k for different ρ and for different distribution shifts.

7.13 Fine-tuning CLIP Models on ImageNet: Extra Details and Figures

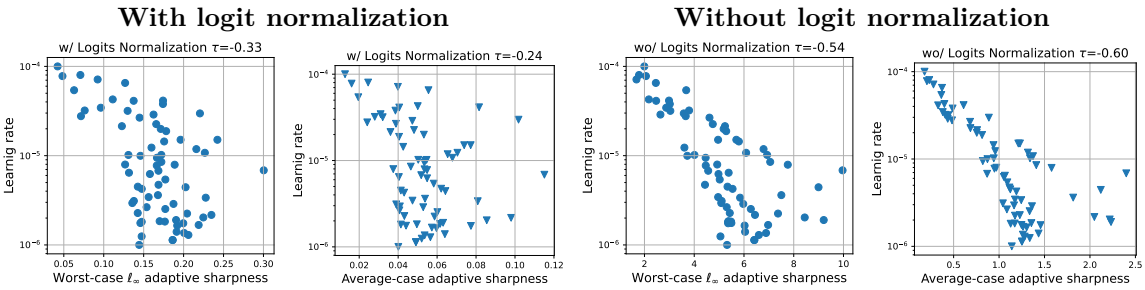


Figure 7.20: Fine-tuning CLIP ViT-B/32 on ImageNet-1k. Sharpness negatively correlates with the size of learning rate for fine-tuning, both with (left) and without (right) using logit normalization. For worst-case sharpness $\rho = 0.002$ is used, for average-case sharpness $\rho = 0.1$.

Experimental details. We take advantage of the models fine-tuned by Wortsman et al. (2022a) from a pre-trained CLIP ViT-B/32, with randomly sampled training hyperparameters (see *random search* setup in Wortsman et al. (2022a)), for which the evaluation of ImageNet validation set and distribution shifts are provided.

Extra figures. For each sharpness definition we show for three values of ρ the correlation between test error on ImageNet (in-distribution) and on the various distribution shifts. In particular, we use worst-case ℓ_∞ adaptive sharpness with (Fig. 7.21) and without (Fig. 7.22) logit normalization, and average-case adaptive sharpness with (Fig. 7.23) and without (Fig. 7.24) logit normalization. For all figures we represent with colors represent the size of the learning rate used for fine-tuning (darker color means larger learning rate). In addition to the datasets shown in Sec. 7.5, we here report the results for ImageNet-V2 (Recht et al., 2019) and ObjectNet (Barbu et al., 2019). ImageNet-V2 consists in a new test set for ImageNet models and is sampled from the same image distribution as the existing validation set: then, the performance of the classifiers on it are highly correlated to that on ImageNet validation set, and ImageNet-V2 cannot be considered a distribution shift in the same sense as the other datasets. In general, we observe that sharpness variants are not predictive of the performance on ImageNet and ImageNet-V2. Moreover, there is in most cases a negative correlation with test error in presence of distribution shifts. We hypothesize that this is related to the influence that the learning rate has on sharpness (see Fig. 7.20), i.e. lower values lead to sharper models.

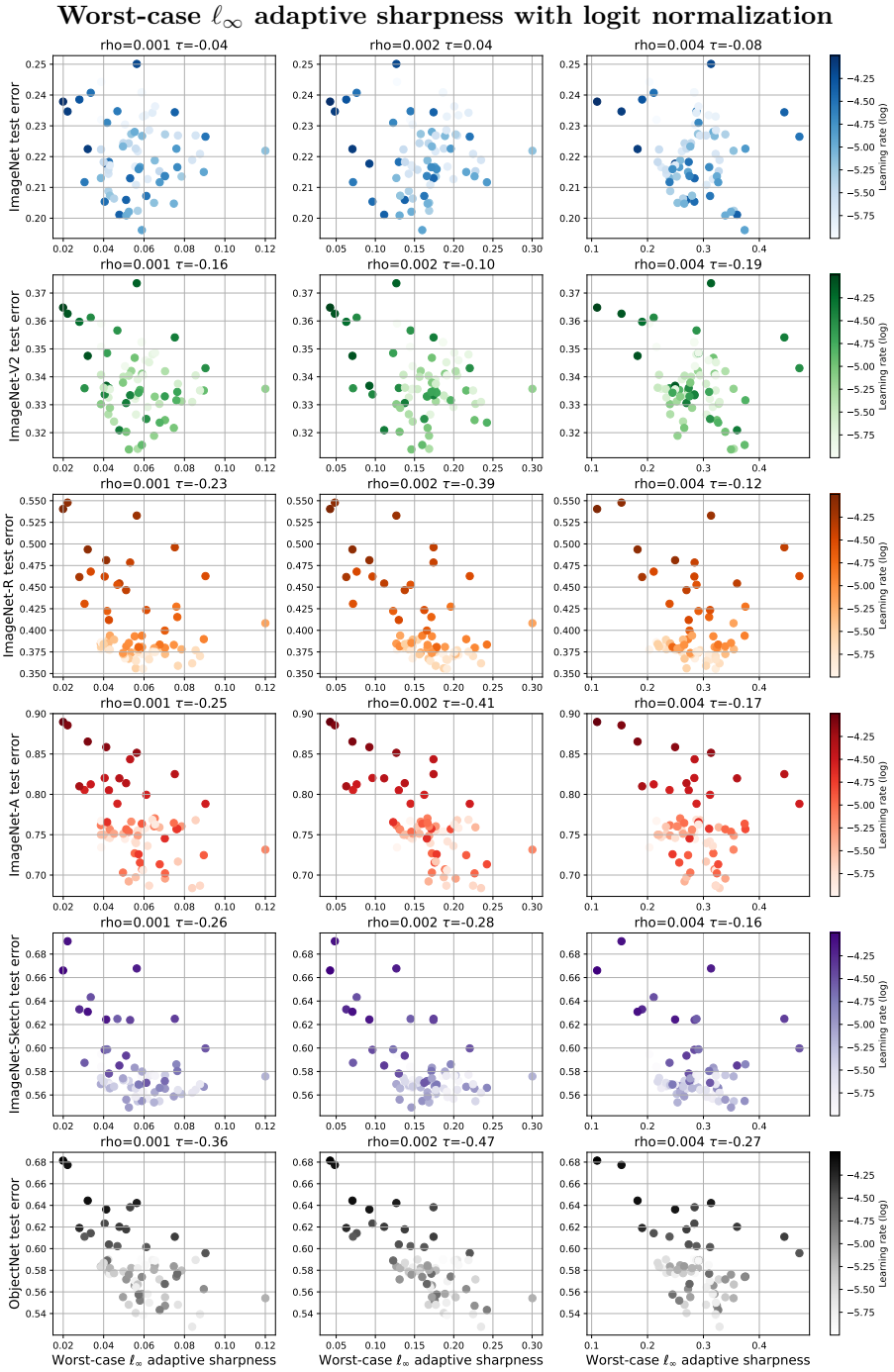


Figure 7.21: Correlation of sharpness with varying ρ with generalization on ImageNet for different distribution shifts.

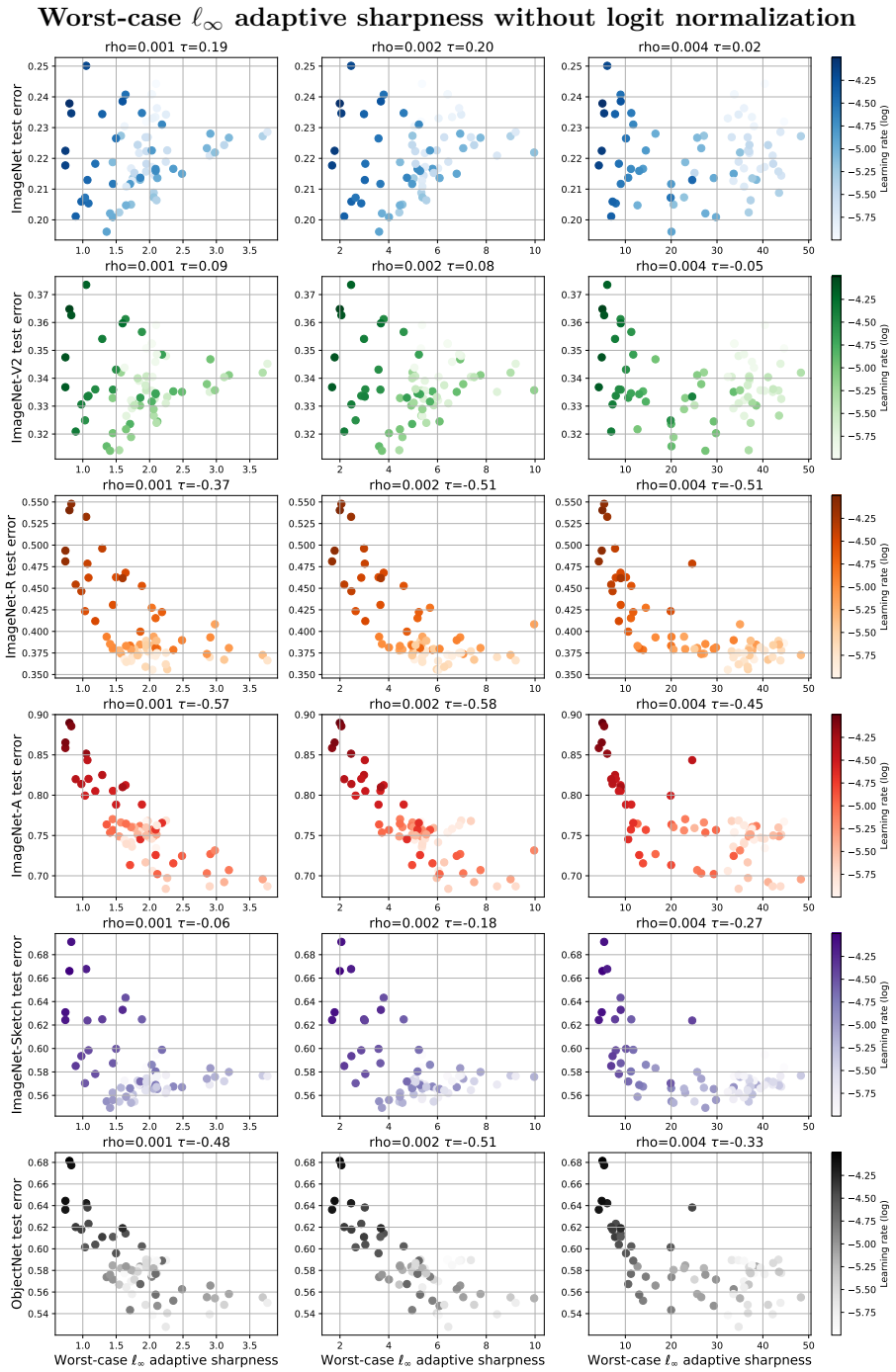


Figure 7.22: Correlation of sharpness with varying ρ with generalization on ImageNet for different distribution shifts.

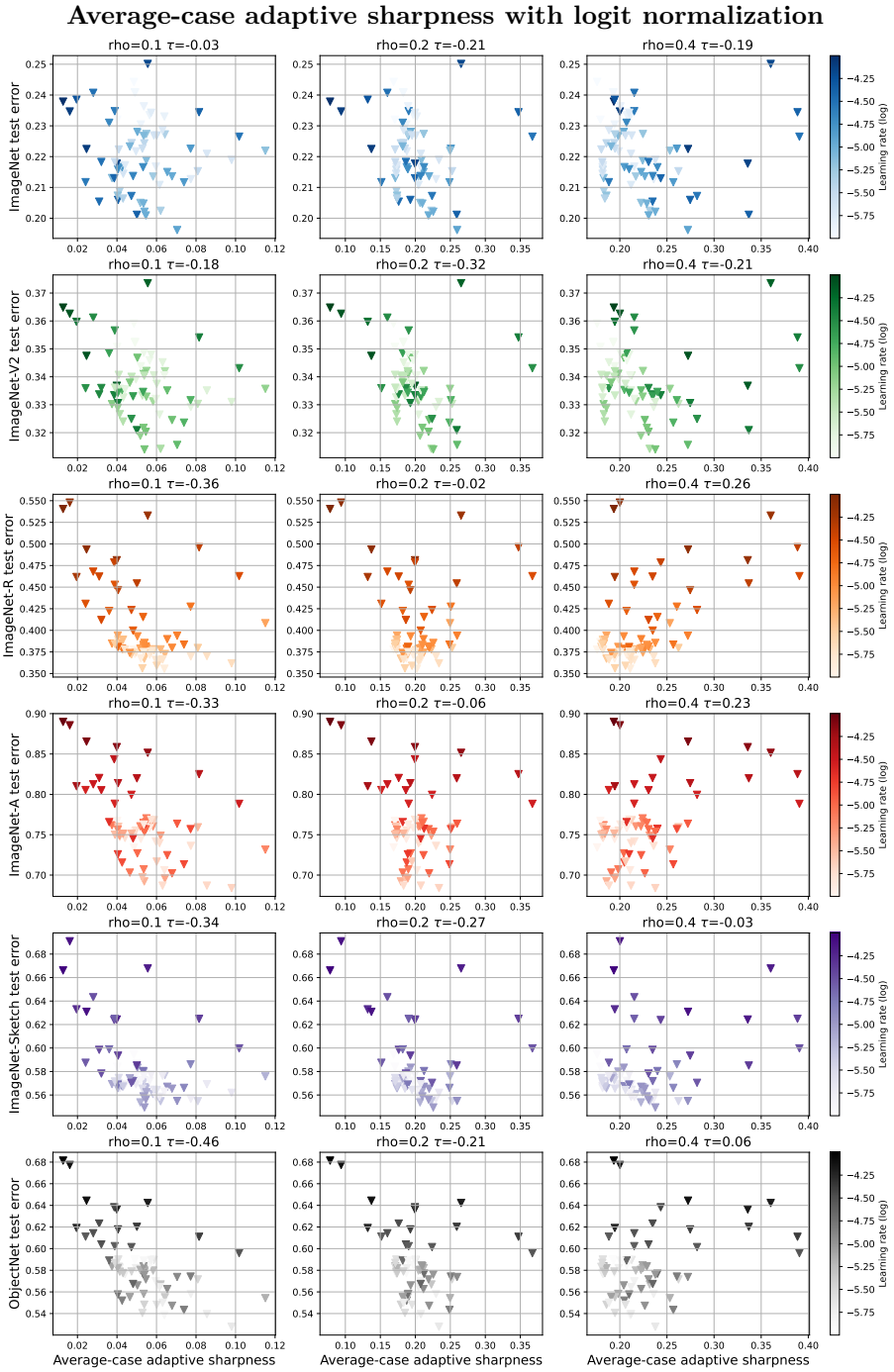


Figure 7.23: Correlation of sharpness with varying ρ with generalization on ImageNet for different distribution shifts.

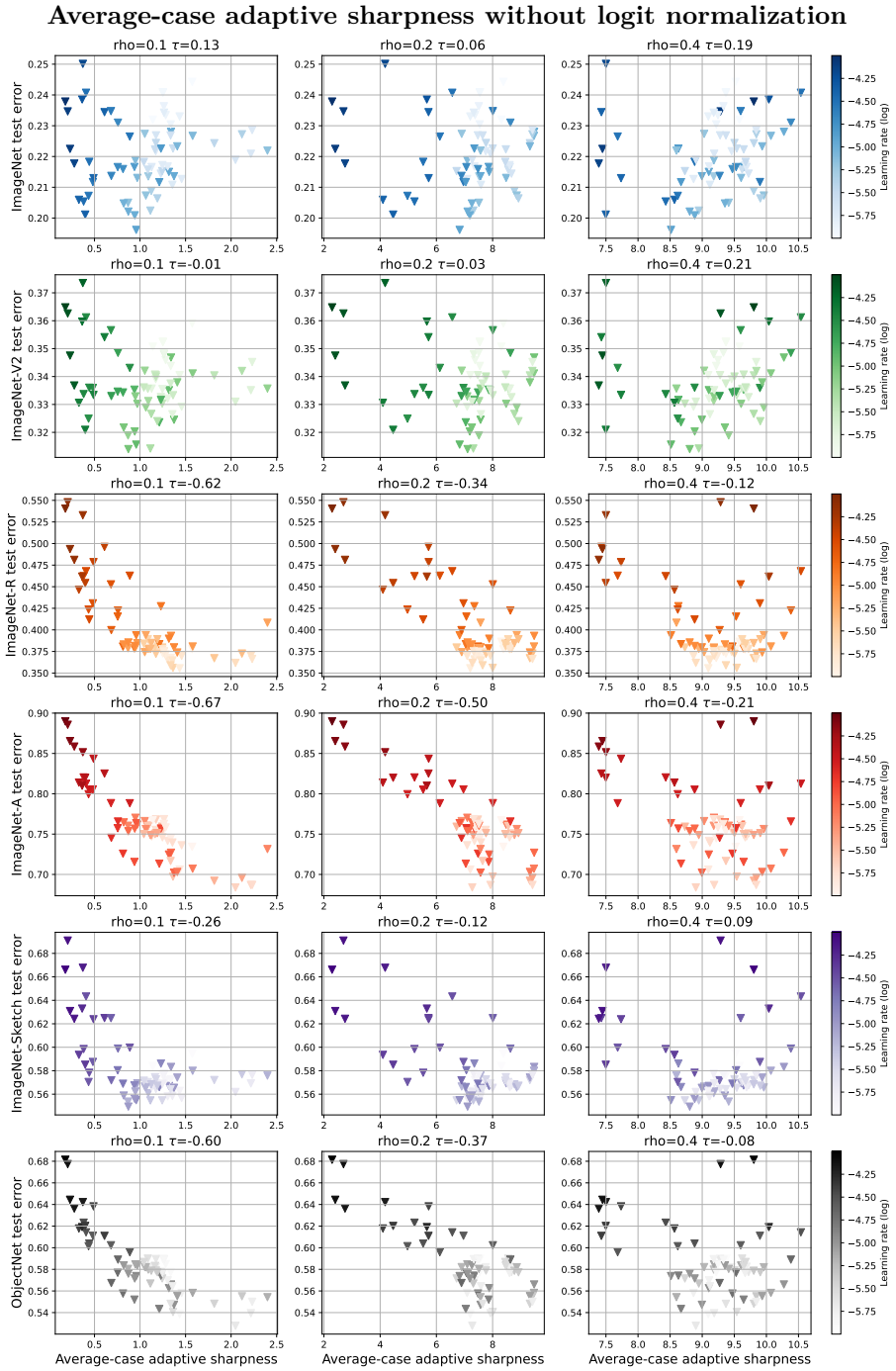


Figure 7.24: Correlation of sharpness with varying ρ with generalization on ImageNet for different distribution shifts.

7.14 Fine-tuning on MNLI: Extra Details and Figures

Experimental details. The models from McCoy et al. (2020) we use are BERT fine-tuned with initialization weights of `bert-case-uncased`. The in-distribution test error is computed on the MNLI `matched` development set, that is a classification task with three classes. As out-of-distribution datasets we use three categories of HANS considered "Inconsistent with heuristic" (see McCoy et al. (2020): *Lexical overlap*, on which the classifiers show the largest variance in test error, *Subsequence* and *Constituent*. In this case, there are only two possible classes.

Extra figures. For each sharpness definition we show for three values of ρ the correlation between test error on MNLI (in-distribution) and on various HANS subsets (out-of-distribution). In particular, we use worst-case ℓ_∞ adaptive sharpness with (Fig. 7.25) and without (Fig. 7.26) logit normalization, and average-case adaptive sharpness with (Fig. 7.27) and without (Fig. 7.28) logit normalization. For all figures we represent with darker colors the models with higher test error on MNLI. In general, all sharpness variants we consider are not predictive of the generalization performance of the model, and in some cases (e.g. Fig. 7.28) there is rather a weak negative correlation between sharpness and test error on out-of-distribution tasks.

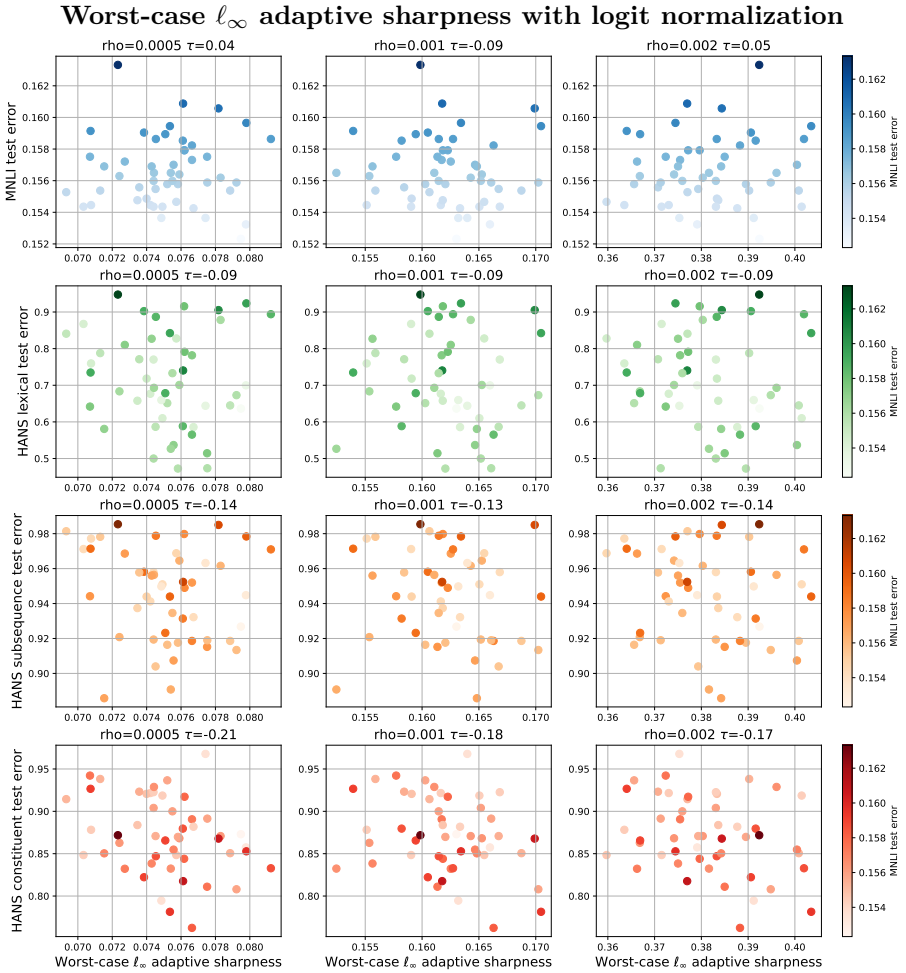


Figure 7.25: Correlation of sharpness with varying ρ with generalization on MNLI for different distribution shifts.

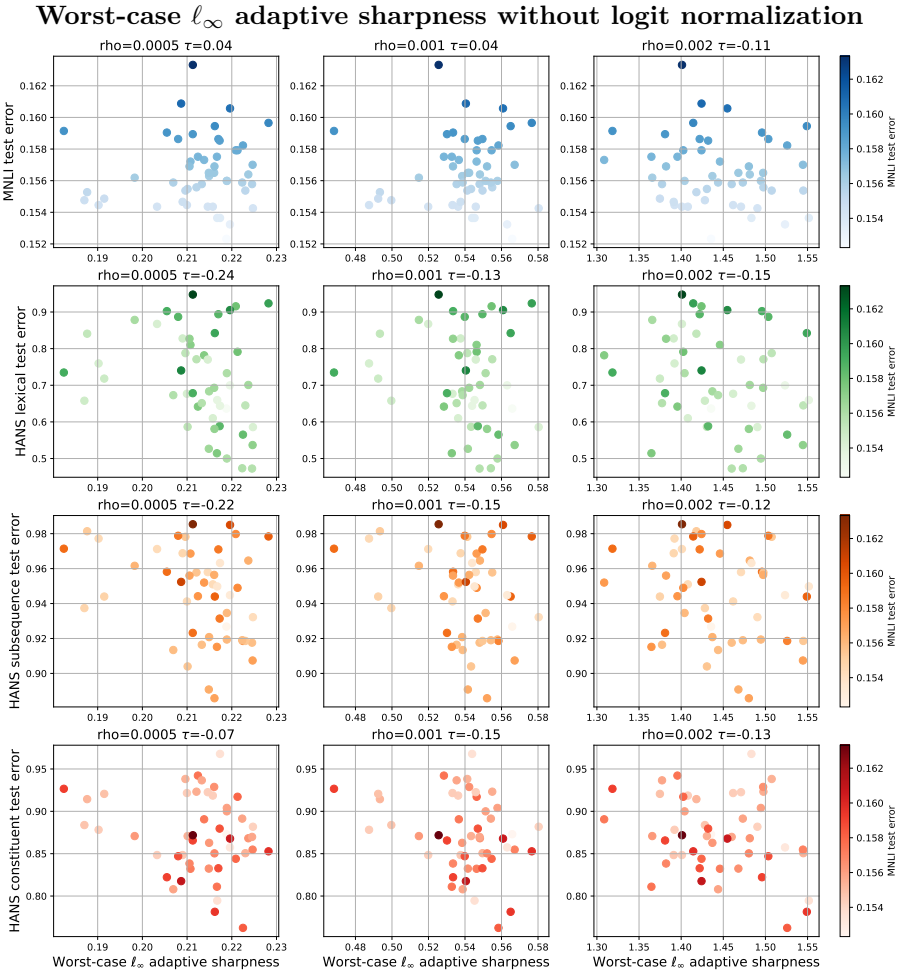


Figure 7.26: Correlation of sharpness with varying ρ with generalization on MNLI for different distribution shifts.

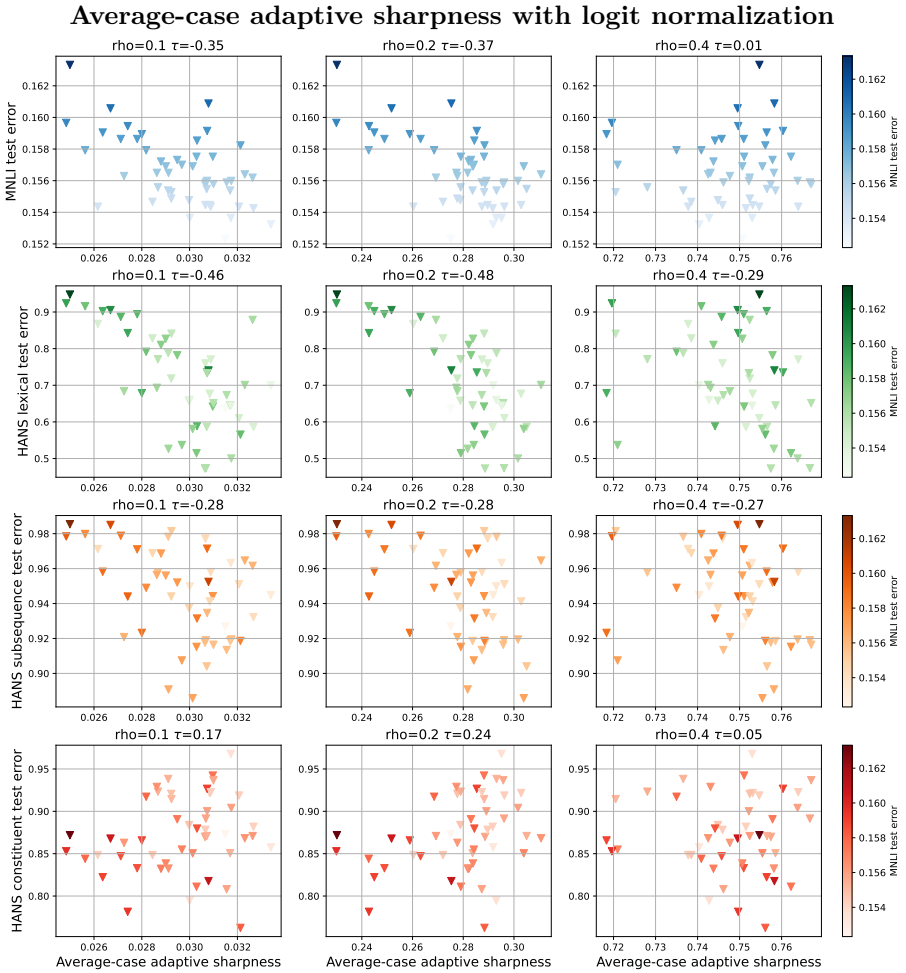


Figure 7.27: Correlation of sharpness with varying ρ with generalization on MNLI for different distribution shifts.

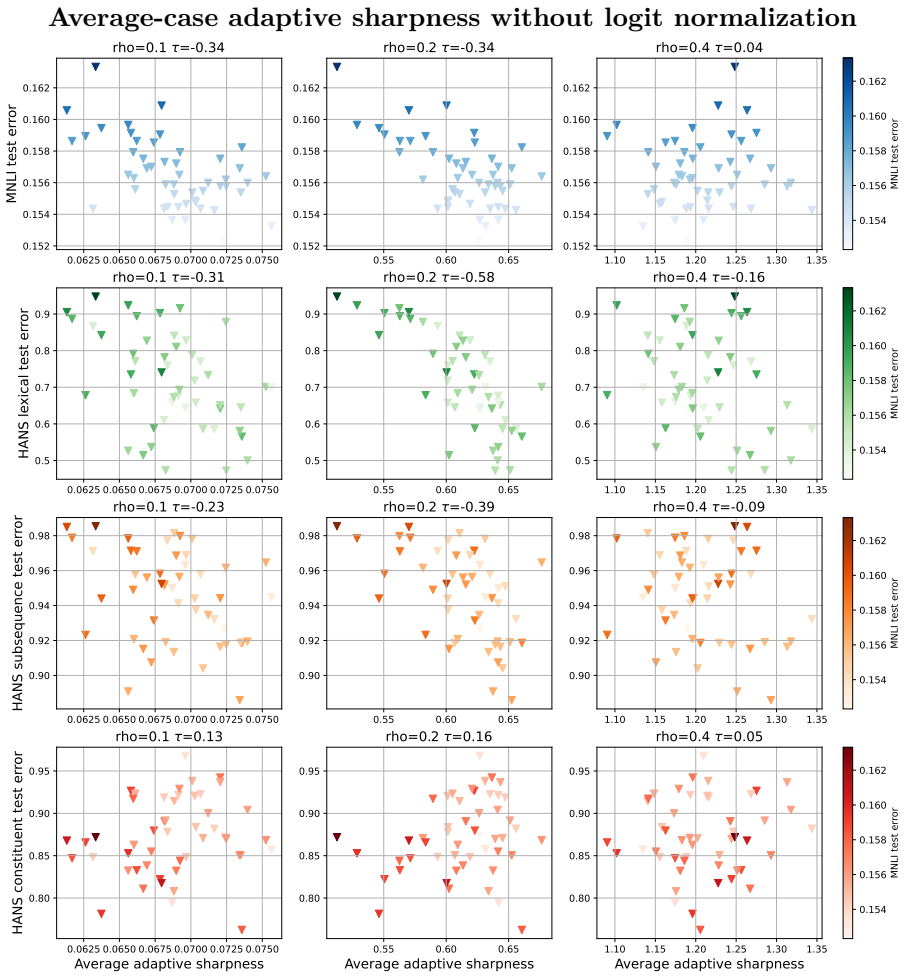


Figure 7.28: Correlation of sharpness with varying ρ with generalization on MNLI for different distribution shifts.

7.15 Training from Scratch on CIFAR-10: Extra Details and Figures

Extra details. We train 200 ResNet-18 and 200 ViT models for 200 epochs using SGD with momentum and linearly decreasing learning rates after a linear warm-up for the first 40% iterations. We found that adding such warm-up to SGD allows us to bridge the gap between SGD and Adam training for ViTs. We use the SimpleViT architecture from the `vit-pytorch` library which is a modification of the standard ViT (Dosovitskiy et al., 2021) with a fixed positional embedding and global average pooling instead of the CLS embedding. We use a ViT model with 4×4 patches, depth of 6 blocks, with 16 heads, embedding size 512, and MLP dimension of 1024. We sample the learning rate from the log-uniform distribution in the range $[0.005, 0.5]$ for ViTs and $[0.05, 5.0]$ for ResNets. We sample uniformly $\rho \in \{0, 0.05, 0.1\}$ of SAM (Foret et al., 2021), with probability 50% mixup ($\alpha = 0.5$) (Zhang et al., 2017b), and with probability 50% standard augmentations combined with RandAugment (with parameters $N = 2$, $M = 14$) (Cubuk et al., 2020). We use $2\times$ repeated augmentations to reduce the augmentation variance from RandAugment (Fort et al., 2021). For CIFAR-10 models, we only show sharpness for well-trained models that have $\leq 1\%$ training error. We note that this selection criterion leaves more ResNets than ViTs on the figures below.

Sharpness evaluation. For sharpness evaluation we use 1024 data points from the training set split in 8 batches: we compute sharpness on each of them and report the average. For worst-case sharpness we use Auto-PGD for 20 steps (for each batch) with random uniform / Gaussian initialization in the feasible set depending on the ℓ_∞ vs. ℓ_2 norm of sharpness. For average-case sharpness, we sample 100 different weights perturbations for every batch.

Extra figures. We present additional figures in Sec. 7.15.1 on the role of data used to evaluate sharpness, in Sec. 7.15.2 on the role of the number of iterations in Auto-PGD to estimate sharpness, in Sec. 7.15.3 on the role of m in m-sharpness, and in Sec. 7.15.4 on the influence of different sharpness definitions and radii on correlation with generalization.

7.15.1 The Role of Data Used for Sharpness Evaluation

We emphasize that for all experiments, we evaluate sharpness on the original training set (CIFAR-10, ImageNet or MNLI) *without augmentations*. However, one may wonder how sensitive this choice is compared to evaluation on the *augmented* training set, particularly in presence of strong data augmentations such as RandAugment (Cubuk et al., 2020) used for training of some models. To test this, in Fig. 7.29, we compare adaptive average-case sharpness computed on the original training set and on augmented training set of CIFAR-10 for ResNets-18. We find that the overall trend is nearly the same for small ρ and differs more strongly for larger ρ where the overall correlation with generalization becomes significantly negative (-0.74 for the largest ρ) on augmented data. In addition,

a side-by-side comparison of sharpness on standard vs. augmented training shows that the relationship between them does not deviate too much from a linear trend, especially when considering separately models trained with and without augmentations.

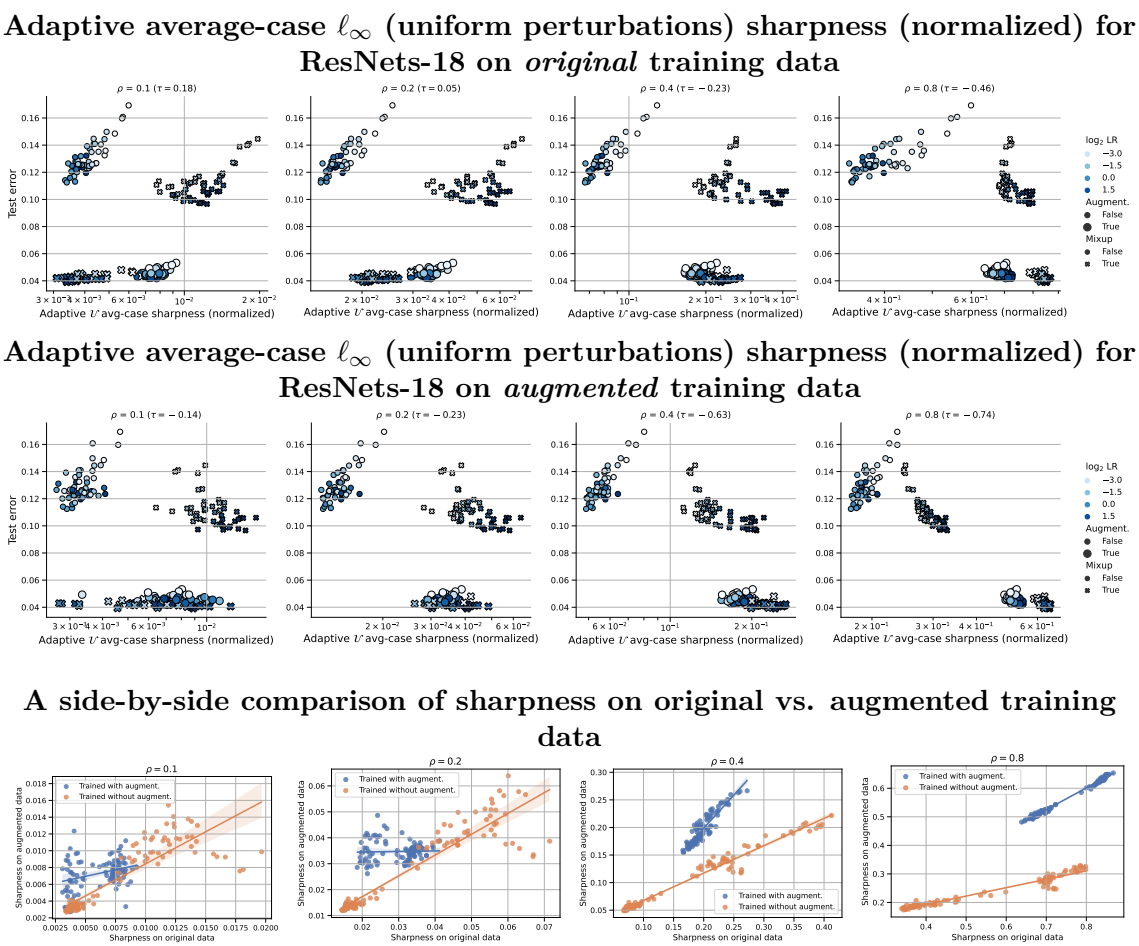
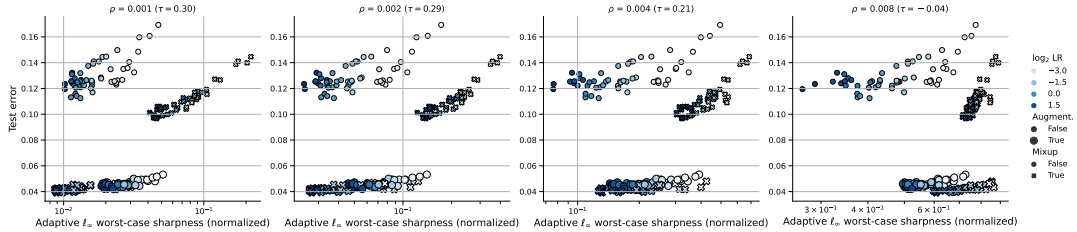


Figure 7.29: Adaptive average-case ℓ_∞ (uniform perturbations) sharpness (normalized) for ResNets-18 on *original* vs. *augmented* CIFAR-10 training data for ResNets-18 for different radii ρ.

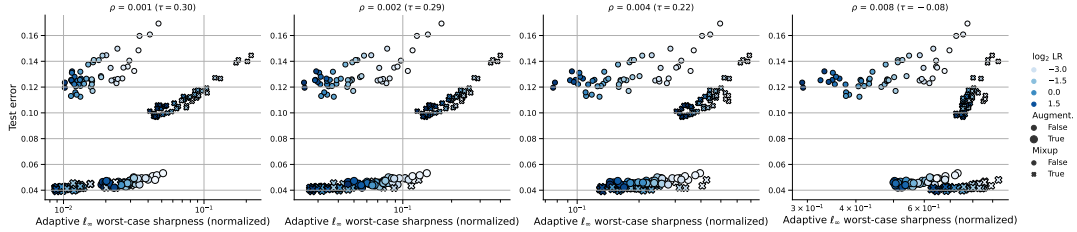
7.15.2 The Role of the Number of Iterations in Auto-PGD

Here we aim to justify the choice of 20 iterations of Auto-PGD in our experiments. In Fig. 7.30, we present results for adaptive worst-case ℓ_∞ sharpness (normalized) for ResNets-18 on CIFAR-10 for 20, 50, 100, and 200 iterations. We can see that the sharpness values are not visibly affected by increasing the number of iterations and the overall trend stays exactly the same.

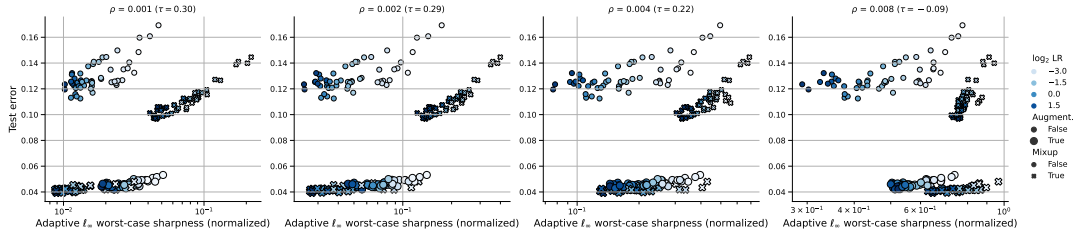
Adaptive worst-case ℓ_∞ sharpness (normalized) for ResNets-18 (20 iterations)



Adaptive worst-case ℓ_∞ sharpness (normalized) for ResNets-18 (50 iterations)



Adaptive worst-case ℓ_∞ sharpness (normalized) for ResNets-18 (100 iterations)



Adaptive worst-case ℓ_∞ sharpness (normalized) for ResNets-18 (200 iterations)

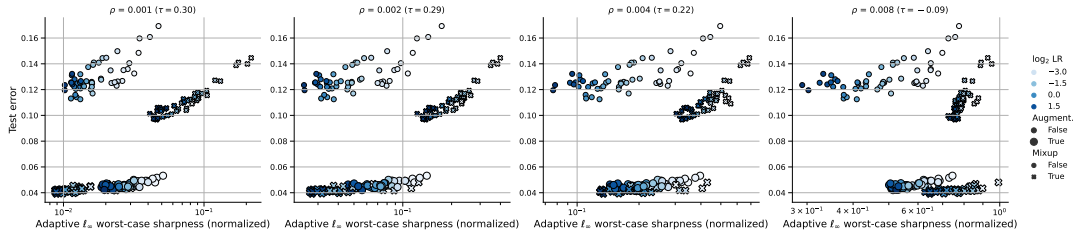


Figure 7.30: Adaptive worst-case ℓ_∞ sharpness (normalized) **for different number of iterations in Auto-PGD** vs. test error on CIFAR-10 for ResNets-18 for different radii ρ.

7.15.3 The Role of m in m-Sharpness

Foret et al. (2021) suggested that a lower m in m-sharpness, i.e., the batch size used for maximizing sharpness, can lead to a higher correlation with generalization in some settings. We note that we have already used a small m for all our experiments ($m = 128$ on CIFAR-10 and $m = 256$ on ImageNet and MNLI), but here we check additionally whether even smaller m change the trend. Fig. 7.31 shows the results sharpness for adaptive worst-case ℓ_∞ sharpness (normalized) for ResNets-18 and ViTs on CIFAR-10 for $m \in \{16, 32, 64, 128\}$. We can see that different m only slightly affects the sharpness values and the overall trend stays unaffected.

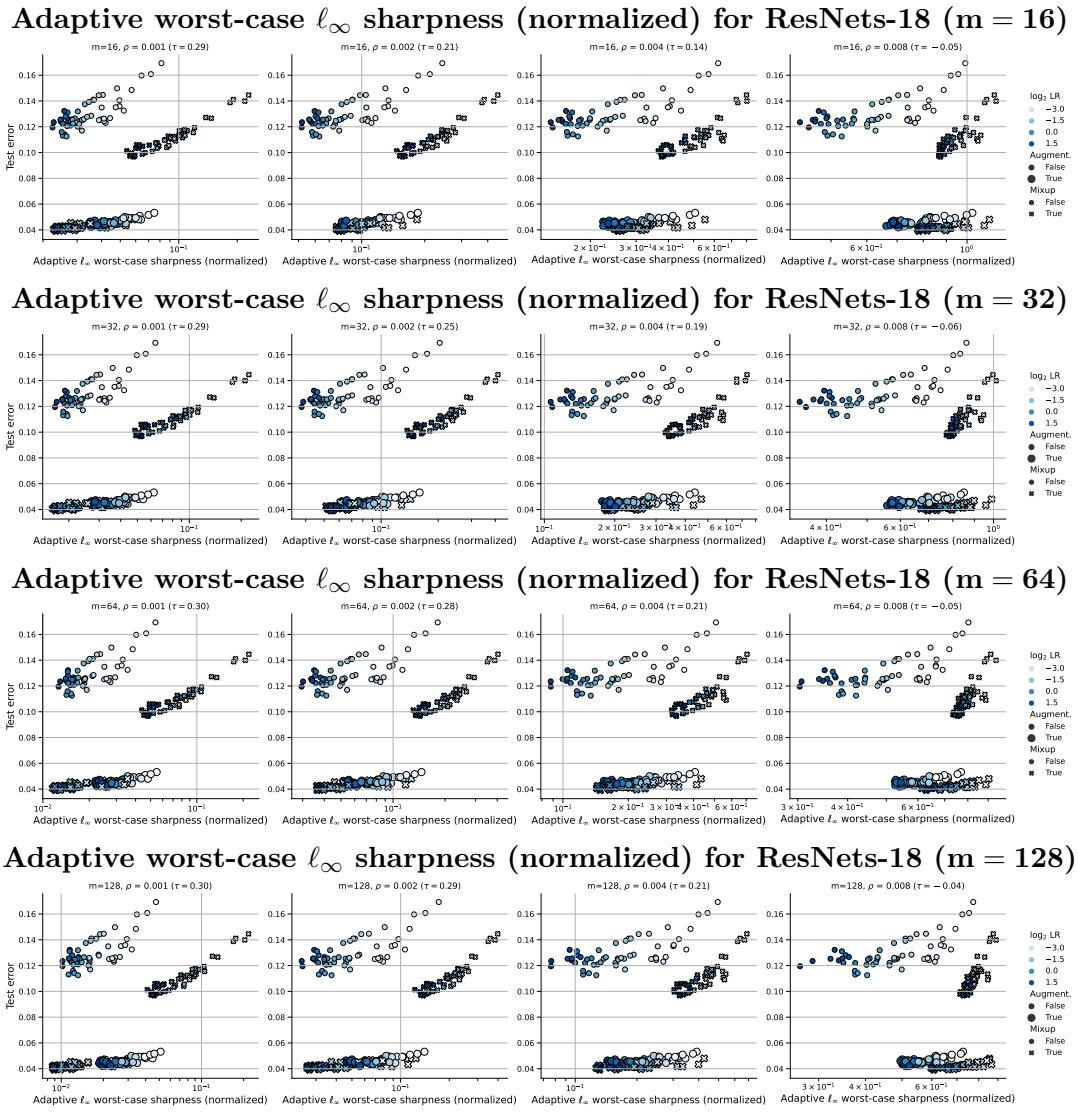


Figure 7.31: Adaptive worst-case ℓ_∞ sharpness (normalized) **for different m** in m-sharpness vs. test error on CIFAR-10 for ResNets-18 for different radii ρ.

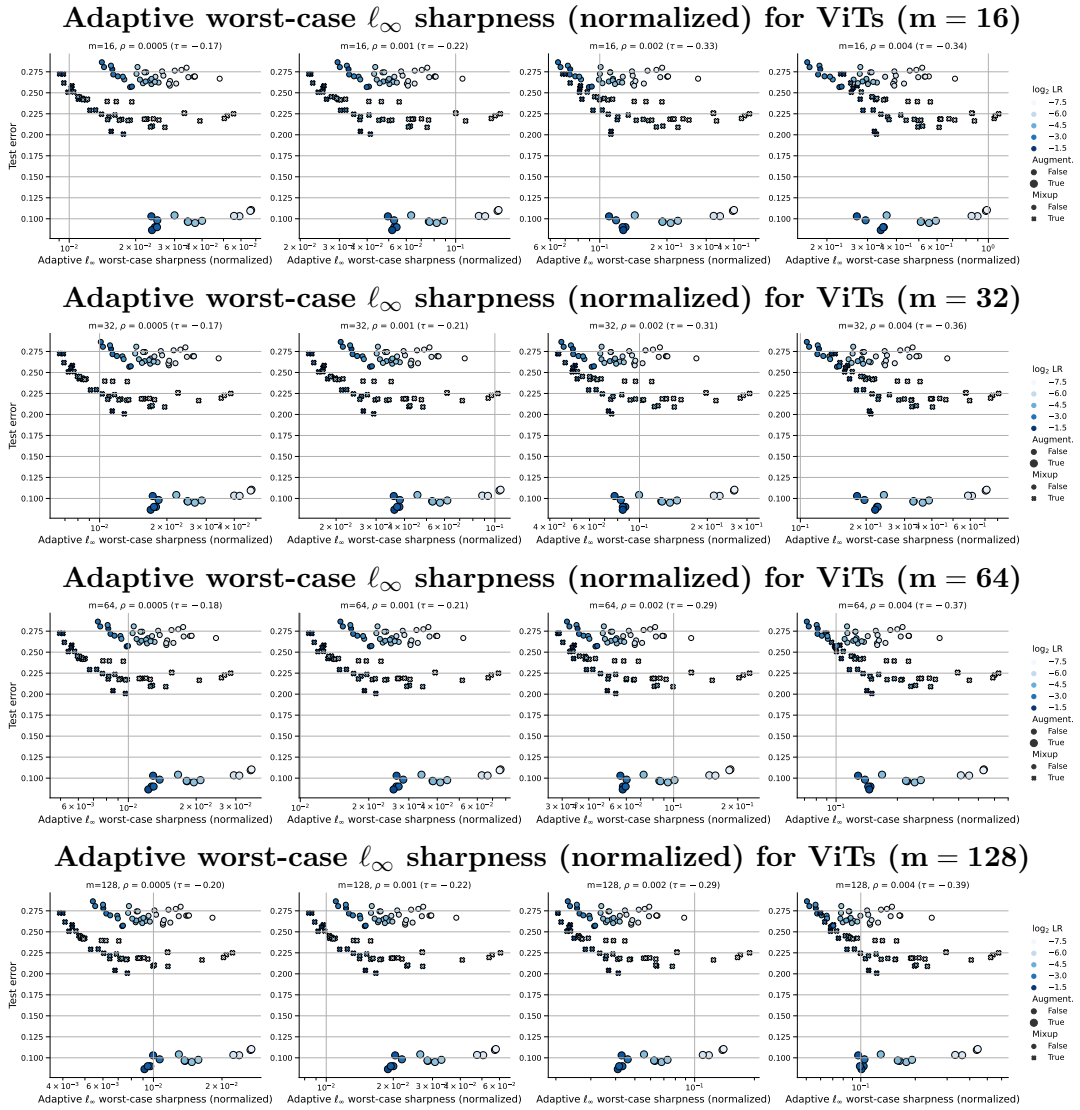


Figure 7.32: Adaptive worst-case ℓ_∞ sharpness (normalized) **for different m** in m-sharpness vs. test error on CIFAR-10 for ViTs for different radii ρ.

7.15.4 The Role of Different Sharpness Definitions and Radii

Here we present results for 12 different sharpness definitions:

- standard average-case ℓ_2 (Gaussian perturbations) sharpness without logit normalization,
- standard worst-case ℓ_2 sharpness without logit normalization,
- adaptive average-case ℓ_2 (Gaussian perturbations) sharpness without logit normalization,
- adaptive worst-case ℓ_2 sharpness without logit normalization,
- adaptive average-case ℓ_2 (Gaussian perturbations) sharpness with logit normalization,
- adaptive worst-case ℓ_2 sharpness with logit normalization,
- standard average-case ℓ_∞ (uniform perturbations) sharpness without logit normalization,
- standard worst-case ℓ_∞ sharpness without logit normalization,
- adaptive average-case ℓ_∞ (uniform perturbations) sharpness without logit normalization,
- adaptive worst-case ℓ_∞ sharpness without logit normalization (*shown in the main part for a single* ρ),
- adaptive average-case ℓ_∞ (uniform perturbations) sharpness with logit normalization,
- adaptive worst-case ℓ_∞ sharpness with logit normalization (*shown in the main part for a single* ρ).

We evaluate a wide range of radii for each sharpness definition to make sure that we do not miss the right scale of sharpness. We present results first for ResNets and then for ViTs.

Observations for ResNets. For ResNets, we observe that many sharpness definitions can successfully capture correlation with standard generalization within each subgroup defined by the values of `augment` $\times$ `mixup`. In particular, on average, *adaptive* sharpness shows a better correlation with generalization within each subgroup, and the best correlation within each subgroup is achieved by ℓ_∞ adaptive worst-case sharpness with logit normalization for a small ρ. In many cases, the correlation of sharpness with OOD generalization on CIFAR-10-C is noticeably lower compared to the correlation of sharpness with standard generalization. Overall, we see that there is no coherent global trend of correlation with generalization that would apply to all models at once. We also observe that for

some sharpness definitions, the flattest models generalize best (for adaptive worst-case ℓ_2 sharpness with normalization for the smallest ρ and for adaptive worst-case ℓ_∞ sharpness without normalization for the largest ρ) but this appears to be unsystematic and there exist nearly equally flat solutions that generalize much worse.

Observations for ViTs. For ViTs, in contrast to ResNets, we do not observe a consistent correlation with generalization even within subgroups. The only exception is the subgroup of points with augmentations where multiple definitions of sharpness tend to correlate with generalization and capture the effect of larger learning rate. We think it is likely due to the fact that with heavy augmentations optimizing the training objective to smaller values is helpful for generalization, while without augmentations all runs have converged within 200 epochs and the learning rate plays no visible role for generalization there. Globally, when taken over all models, the correlation with standard generalization is close to 0 and tends to slightly decrease when we measure OOD generalization on CIFAR-10-C. Finally, we note that there are no cases where the flattest ViT models achieve the best generalization. Thus, even our weak hypothesis about the role of sharpness is not confirmed here.

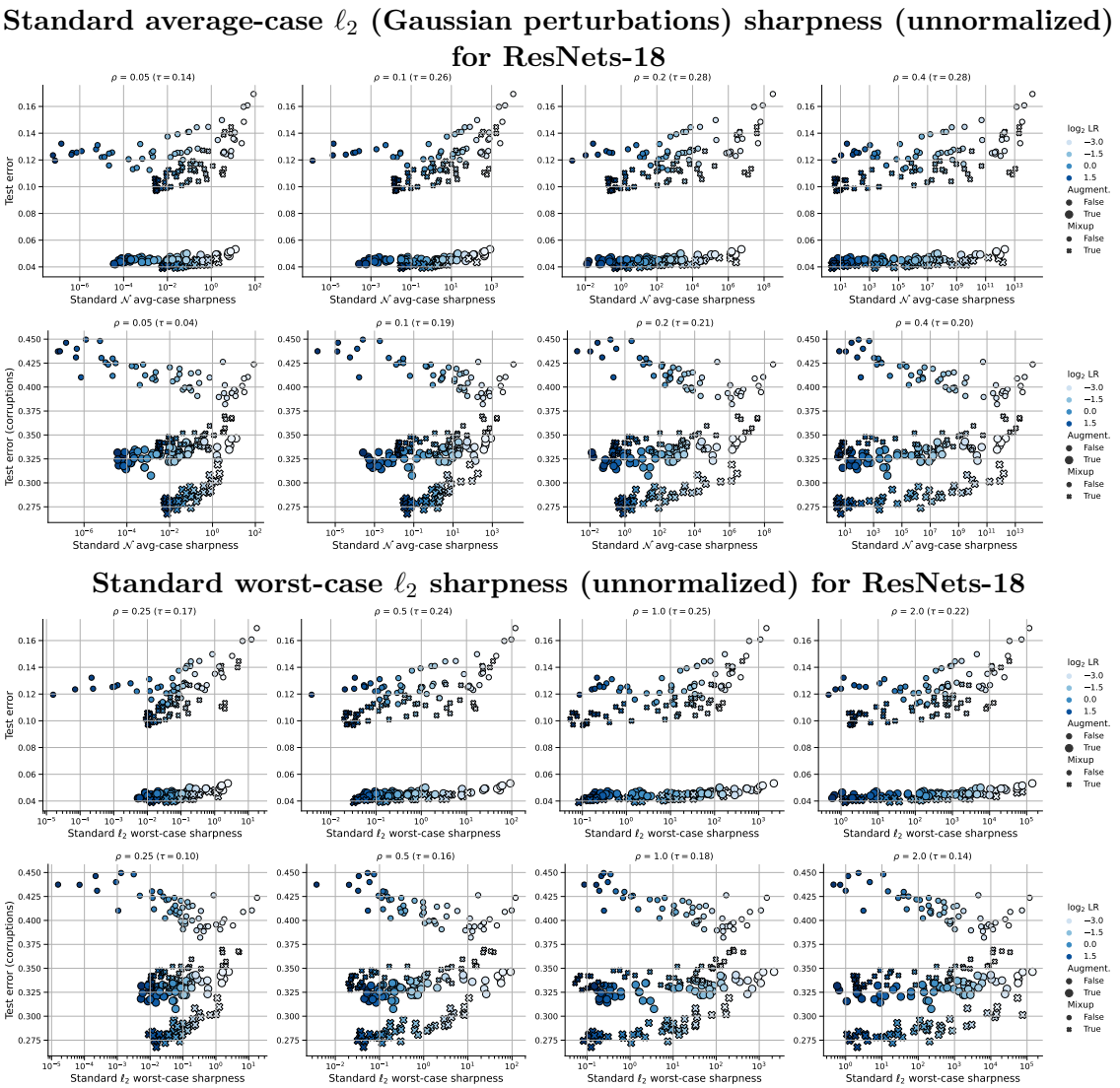


Figure 7.33: Average and worst-case ℓ_2 standard sharpness definitions (unnormalized) vs. test error and OOD test error (common corruptions) on CIFAR-10 for ResNets-18 for different radii ρ.

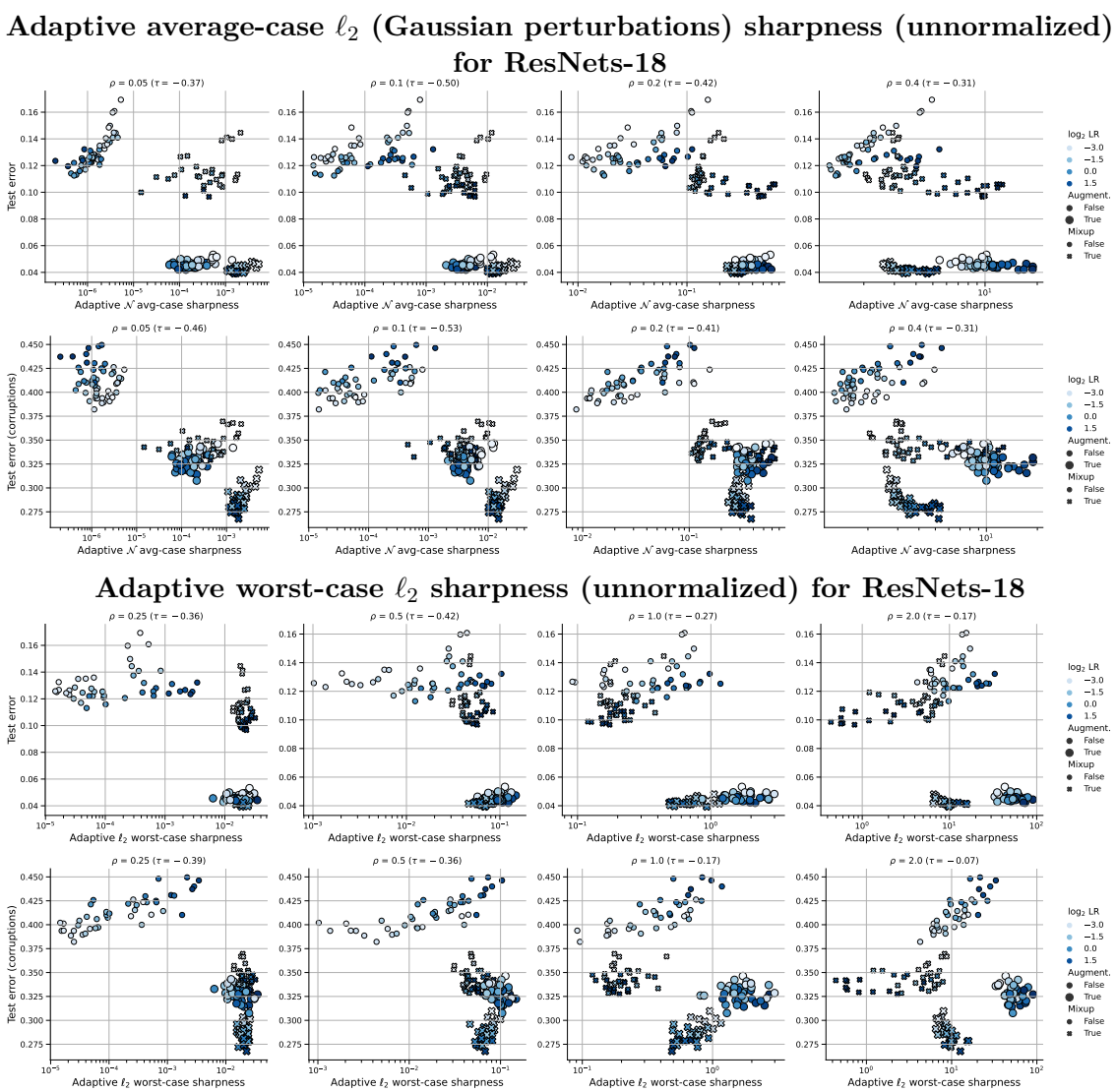


Figure 7.34: Average and worst-case ℓ_2 adaptive sharpness definitions (unnormalized) vs. test error and OOD test error (common corruptions) on CIFAR-10 for ResNets-18 for different radii ρ.

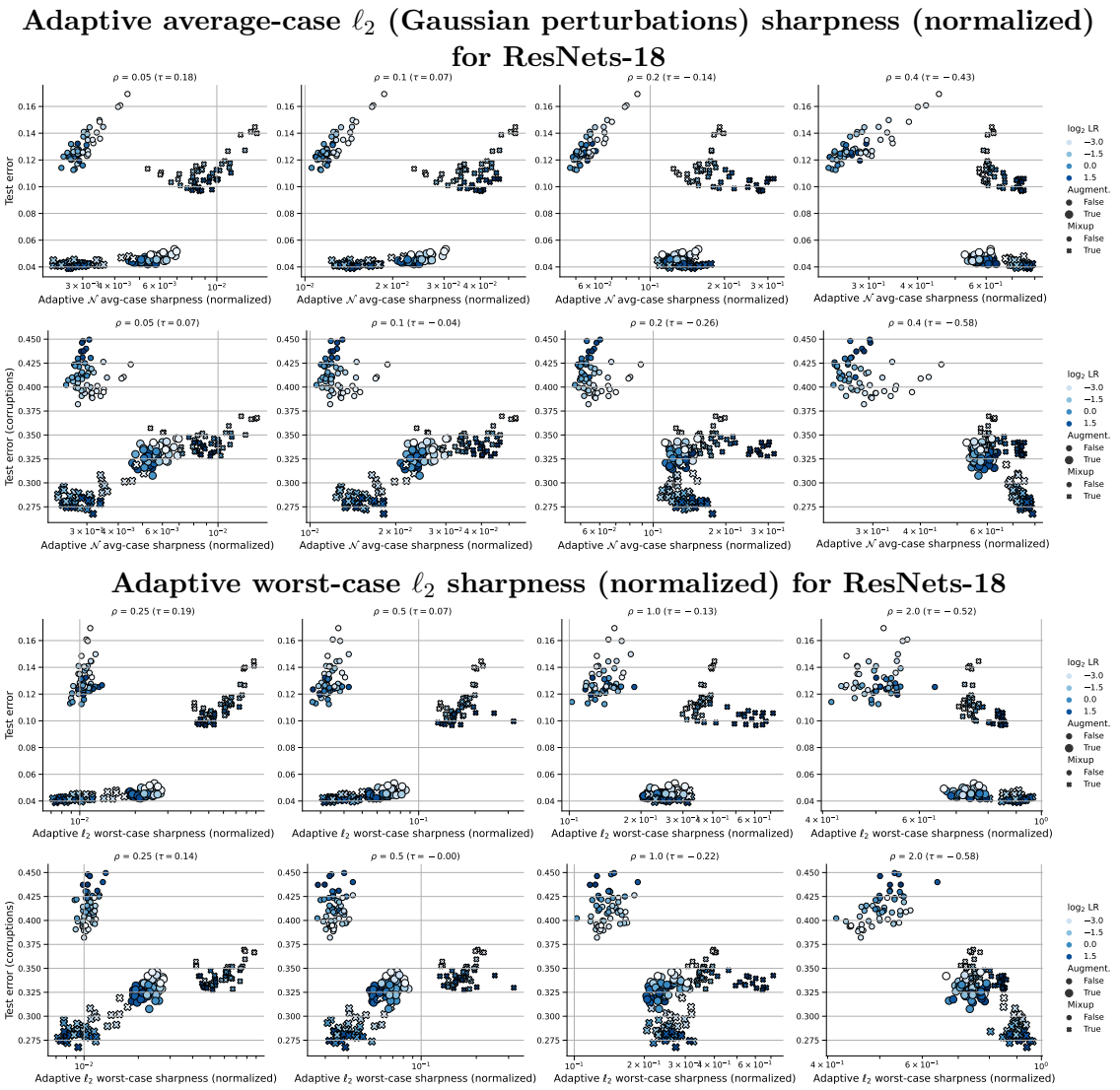


Figure 7.35: Average and worst-case ℓ_2 adaptive sharpness definitions (normalized) vs. test error and OOD test error (common corruptions) on CIFAR-10 for ResNets-18 for different radii ρ.

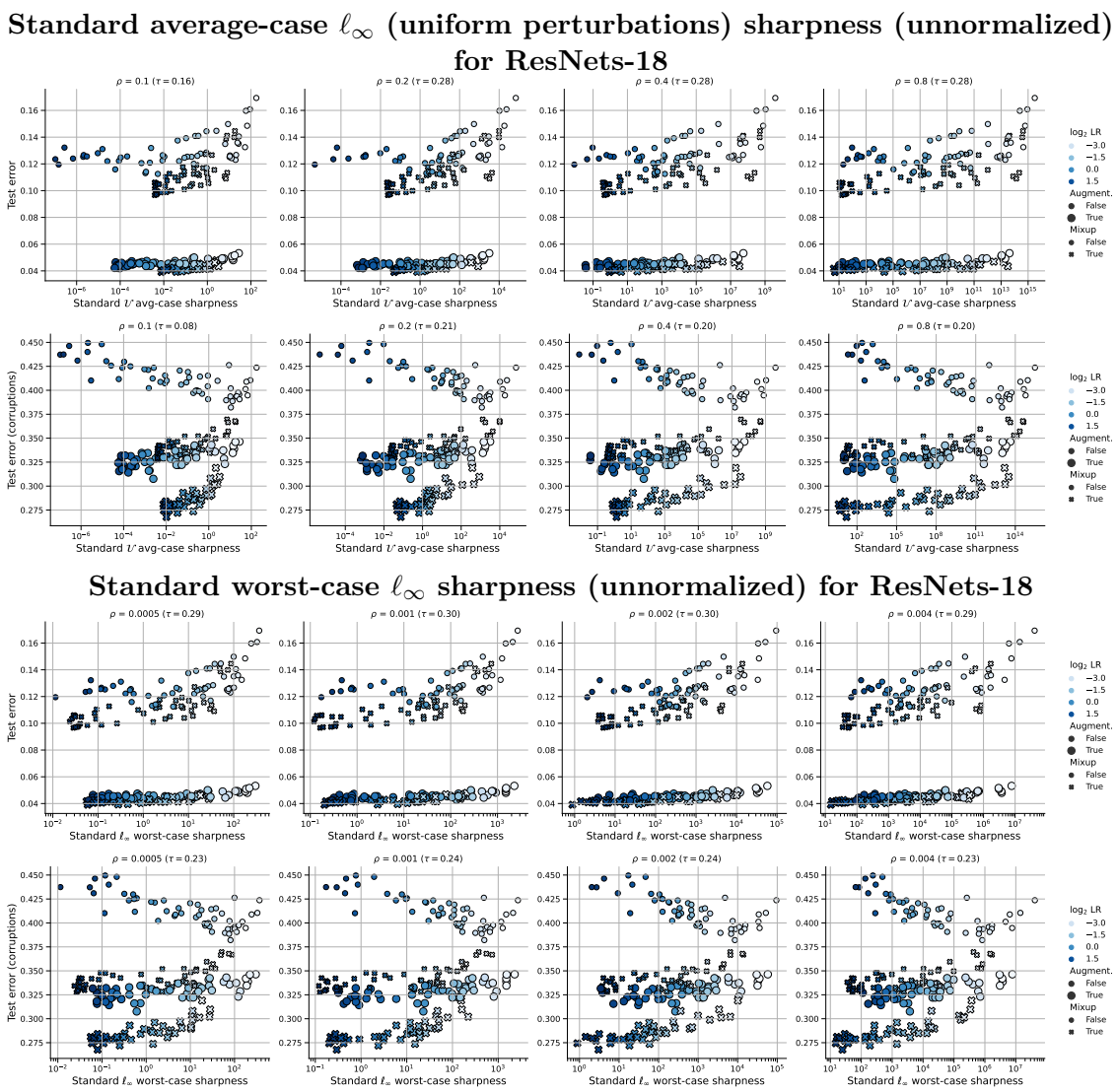


Figure 7.36: Average and worst-case ℓ_∞ standard sharpness definitions (unnormalized) vs. test error and OOD test error (common corruptions) on CIFAR-10 for ResNets-18 for different radii ρ.

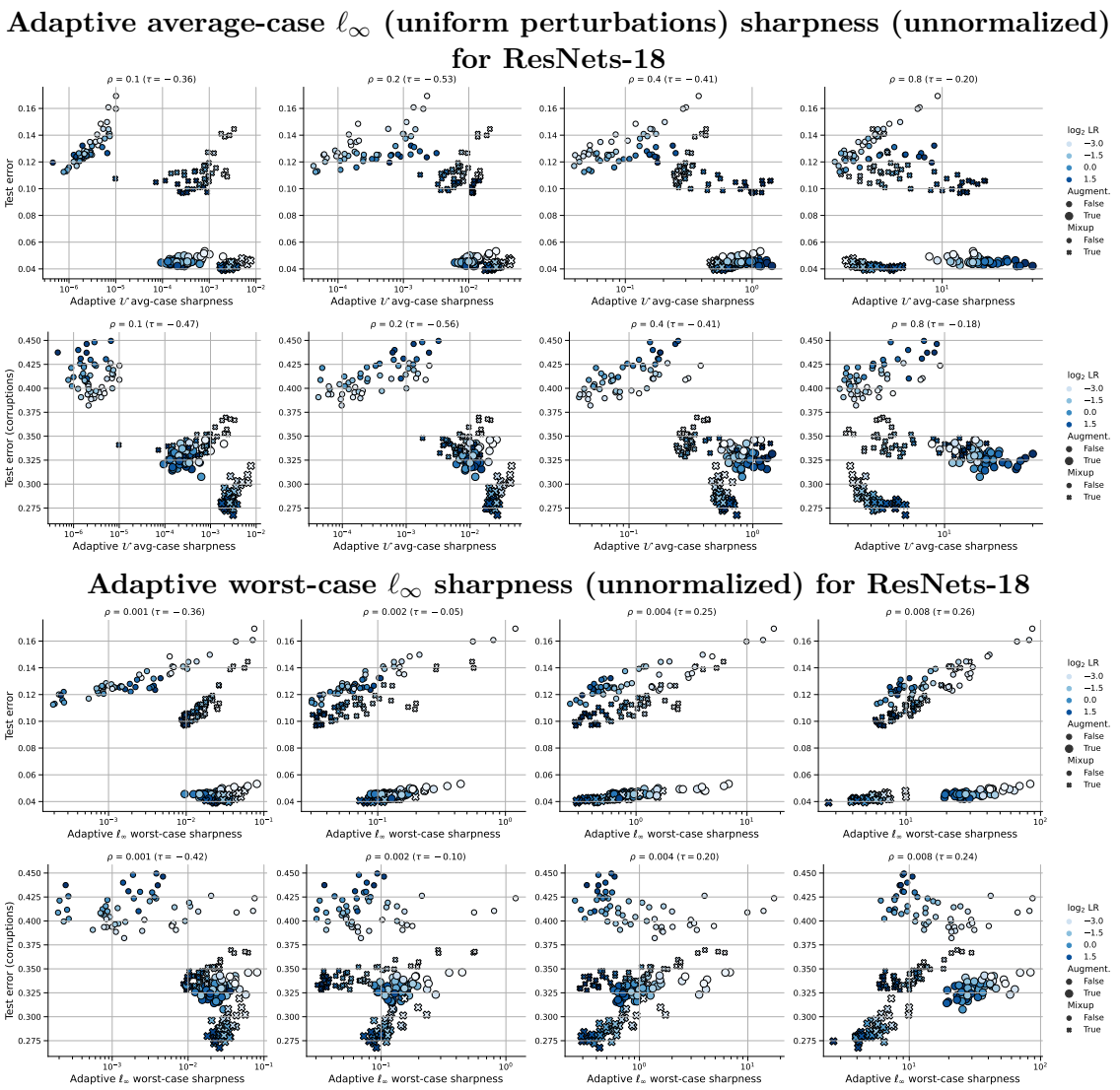


Figure 7.37: Average and worst-case ℓ_∞ adaptive sharpness definitions (unnormalized) vs. test error and OOD test error (common corruptions) on CIFAR-10 for ResNets-18 for different radii ρ.

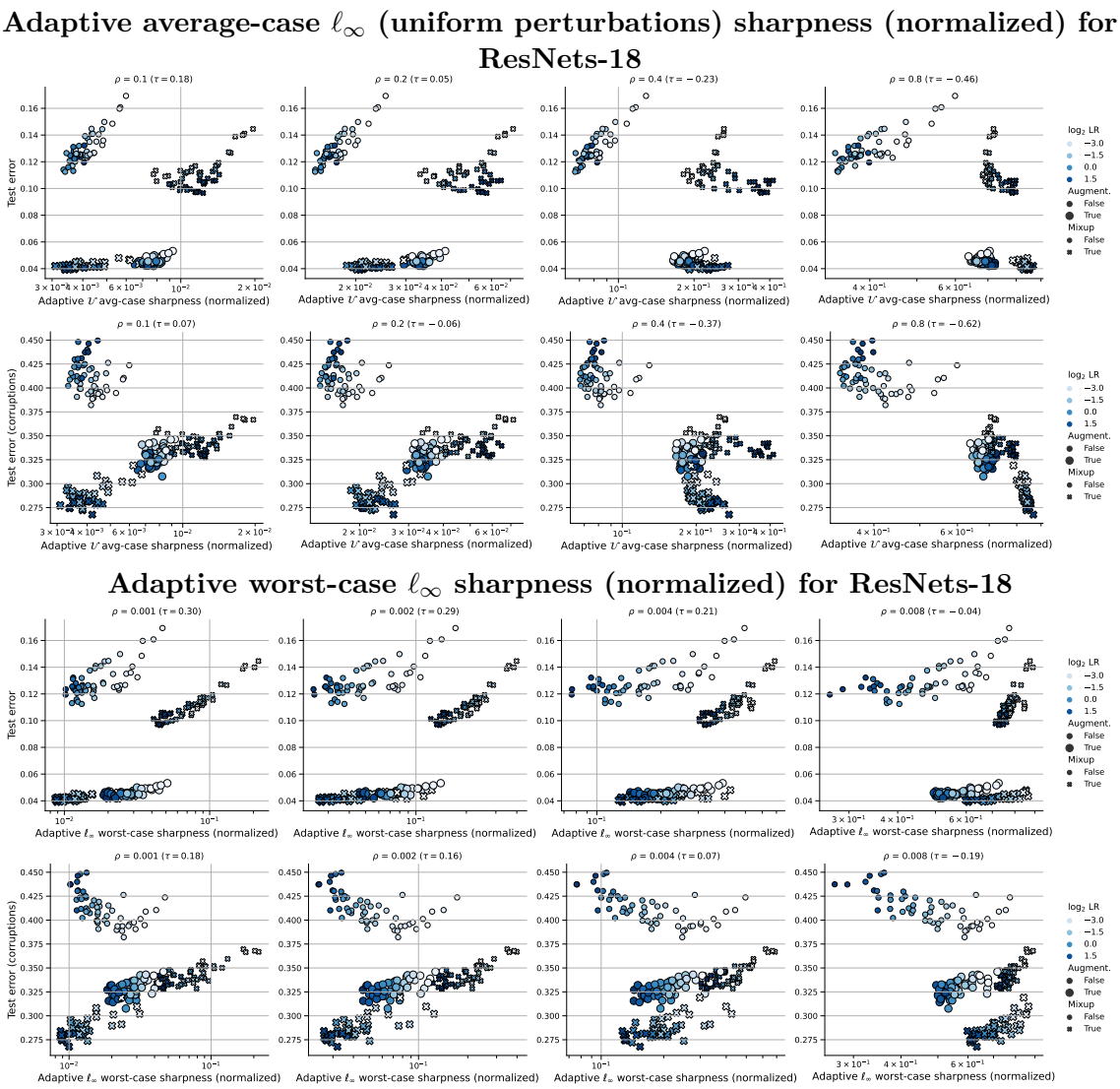


Figure 7.38: Average and worst-case ℓ_∞ adaptive sharpness definitions (normalized) vs. test error and OOD test error (common corruptions) on CIFAR-10 for ResNets-18 for different radii ρ.

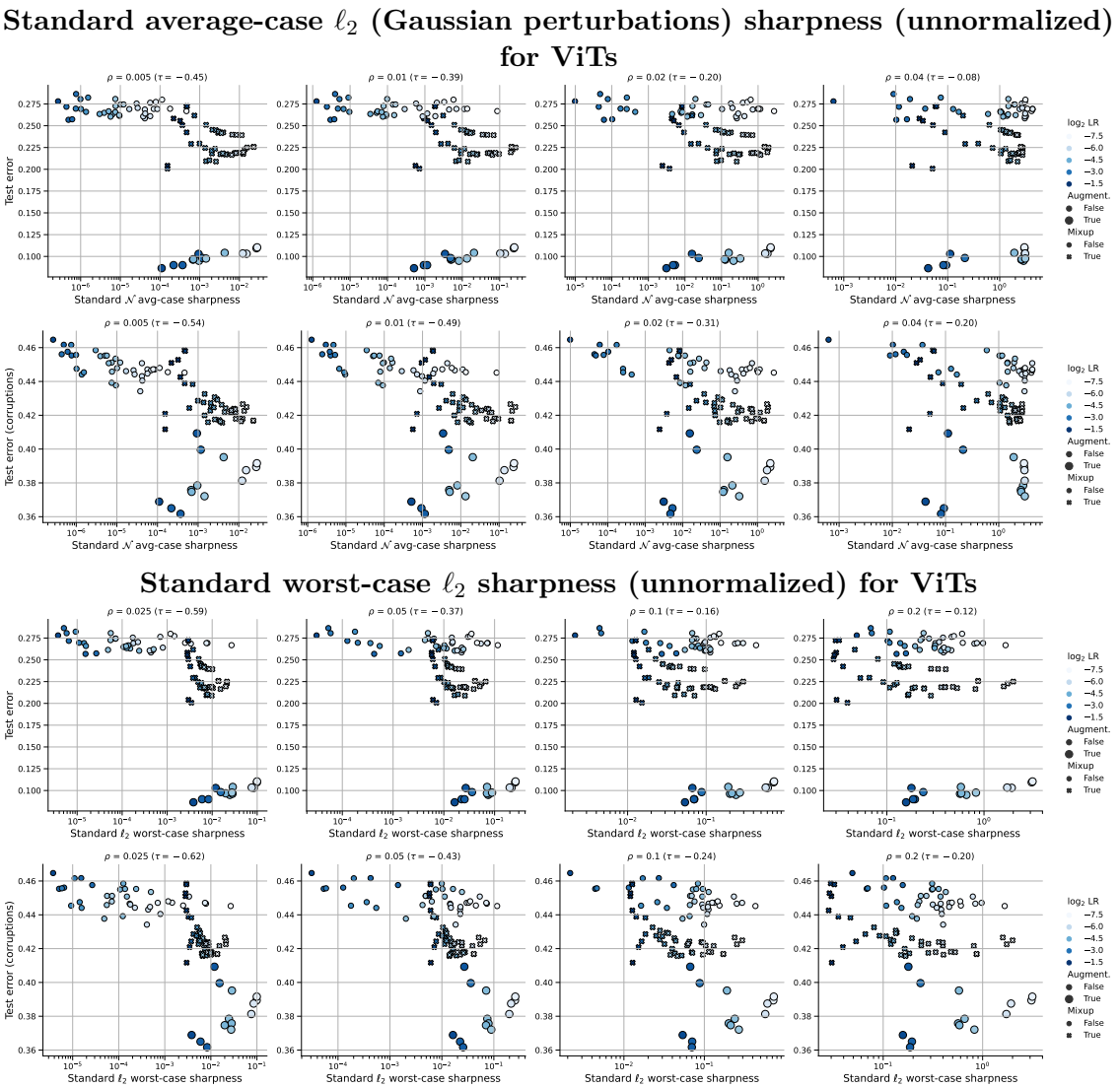


Figure 7.39: Average and worst-case ℓ_2 standard sharpness definitions (unnormalized) vs. test error and OOD test error (common corruptions) on CIFAR-10 for ViTs for different radii ρ.

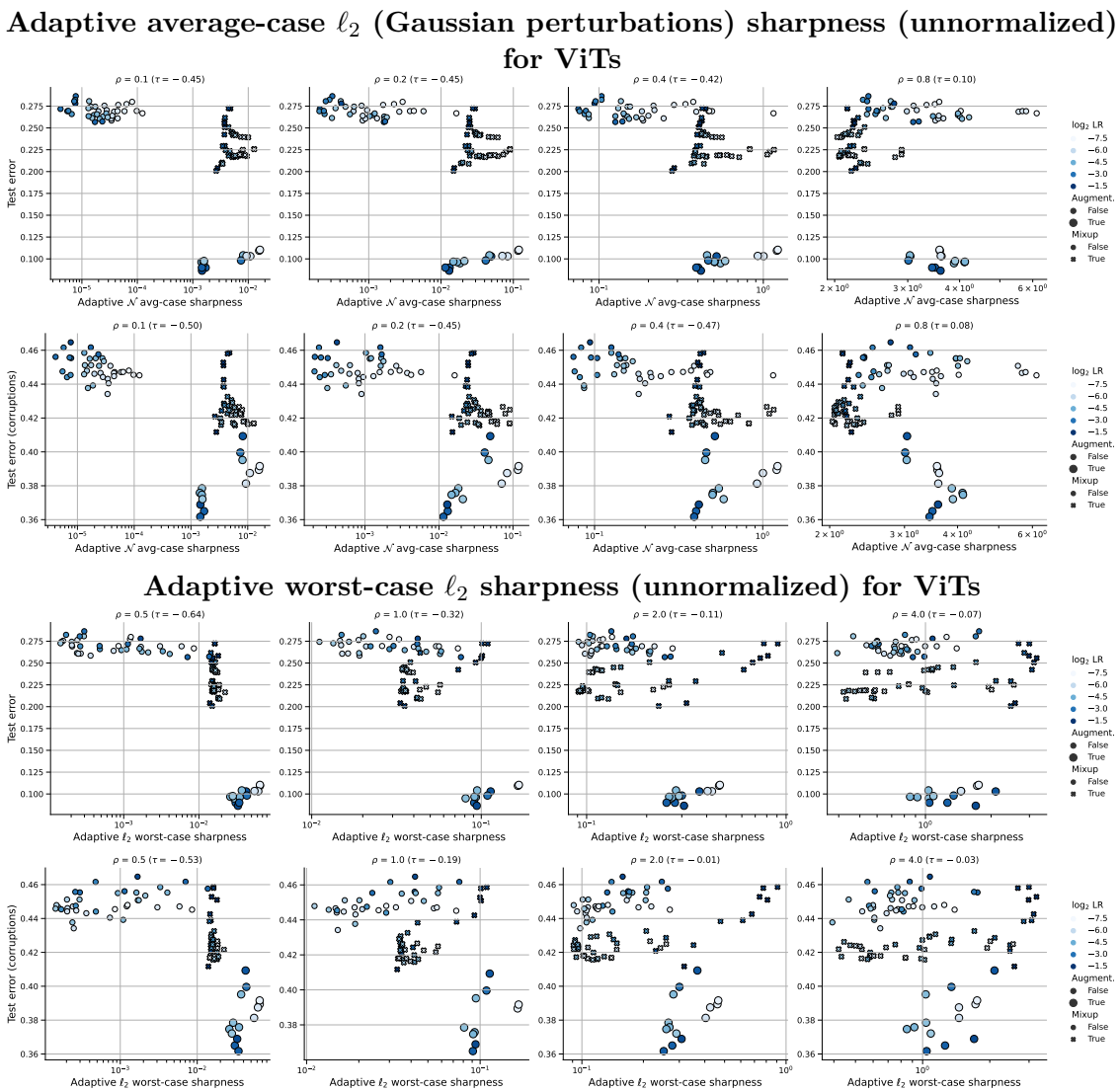


Figure 7.40: Average and worst-case ℓ_2 adaptive sharpness definitions (unnormalized) vs. test error and OOD test error (common corruptions) on CIFAR-10 for ViTs for different radii ρ.

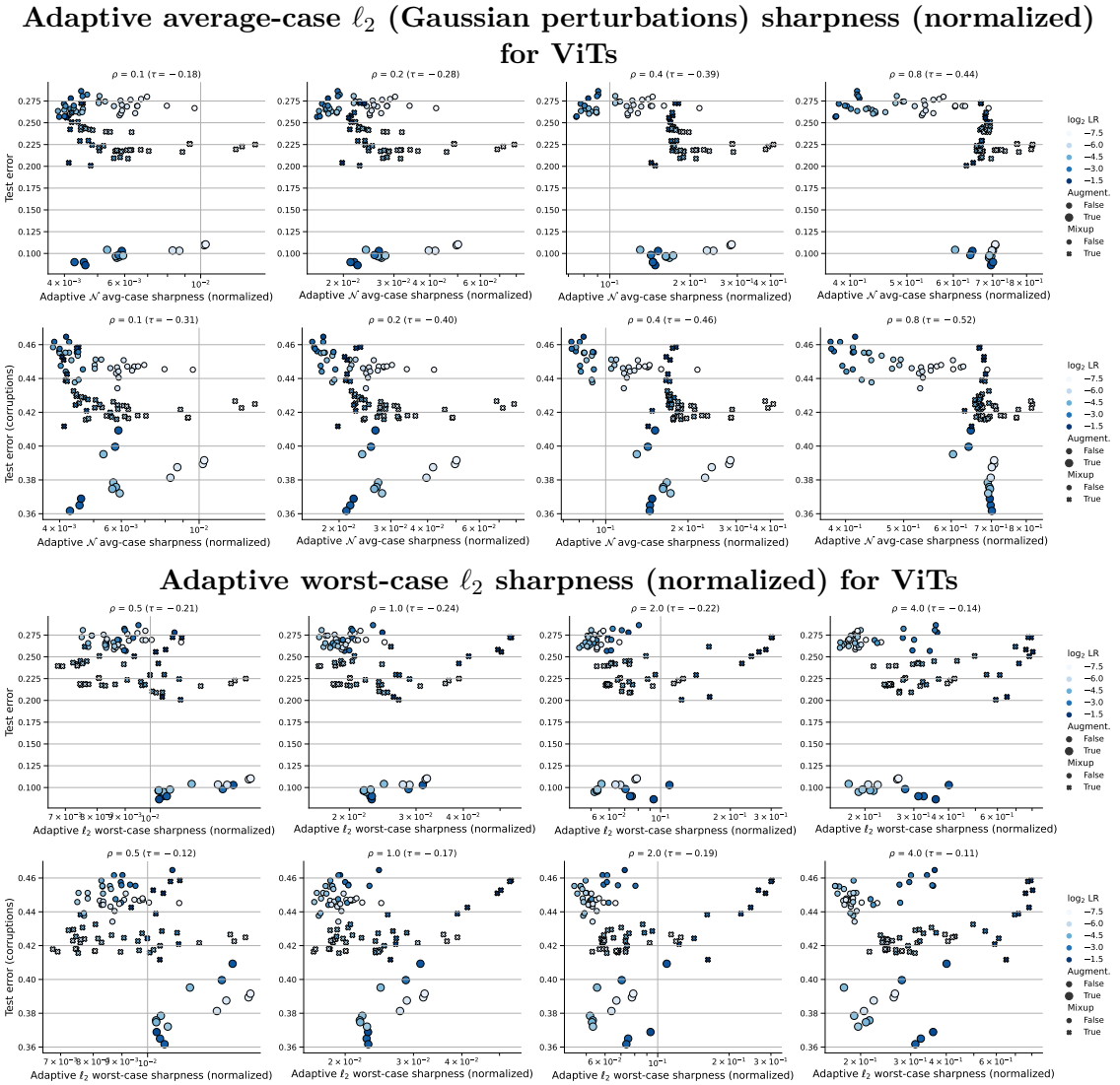


Figure 7.41: Average and worst-case ℓ_2 adaptive sharpness definitions (normalized) vs. test error and OOD test error (common corruptions) on CIFAR-10 for ViTs for different radii ρ.

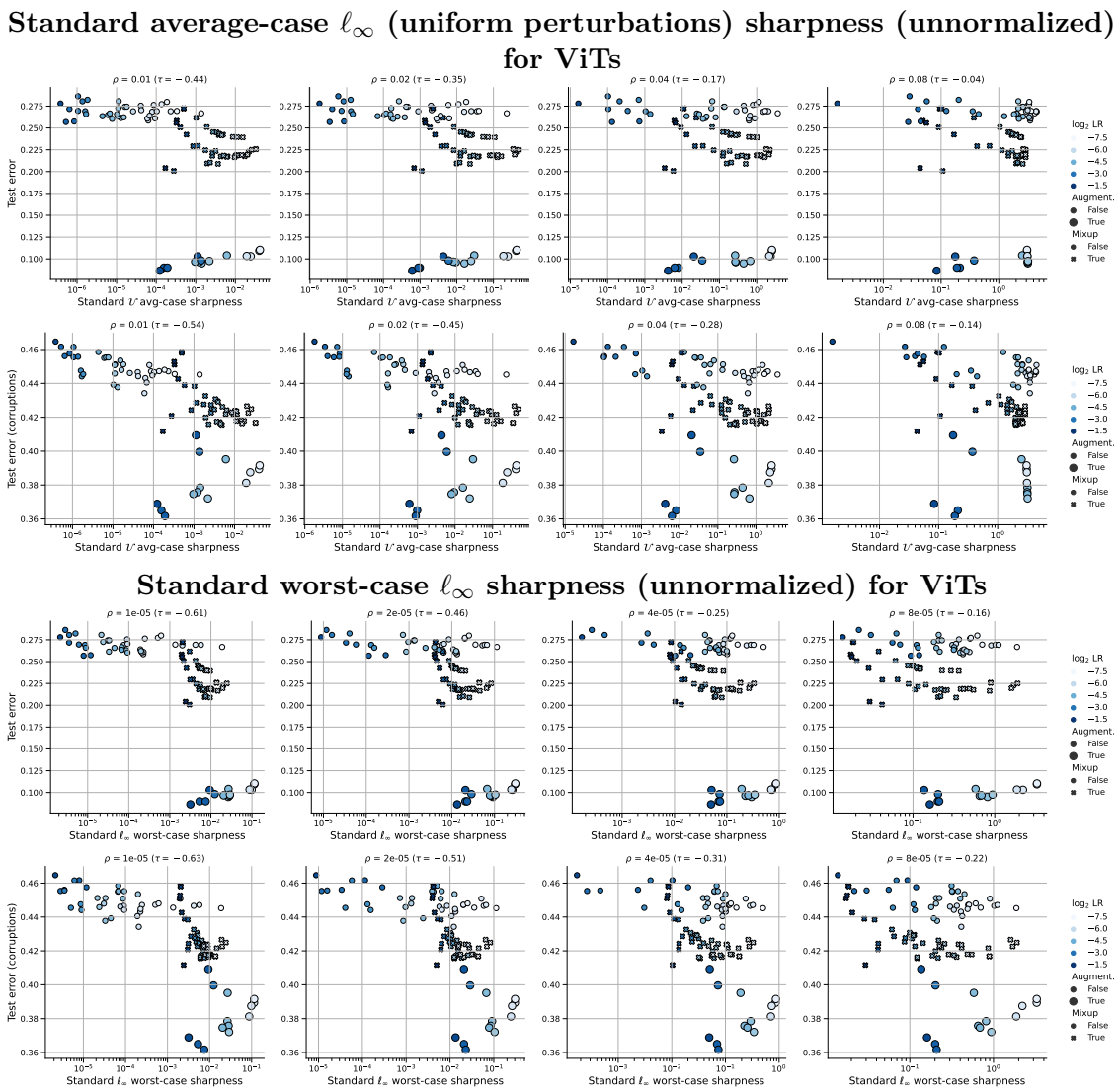


Figure 7.42: Average and worst-case ℓ_∞ standard sharpness definitions (unnormalized) vs. test error and OOD test error (common corruptions) on CIFAR-10 for ViTs for different radii ρ.

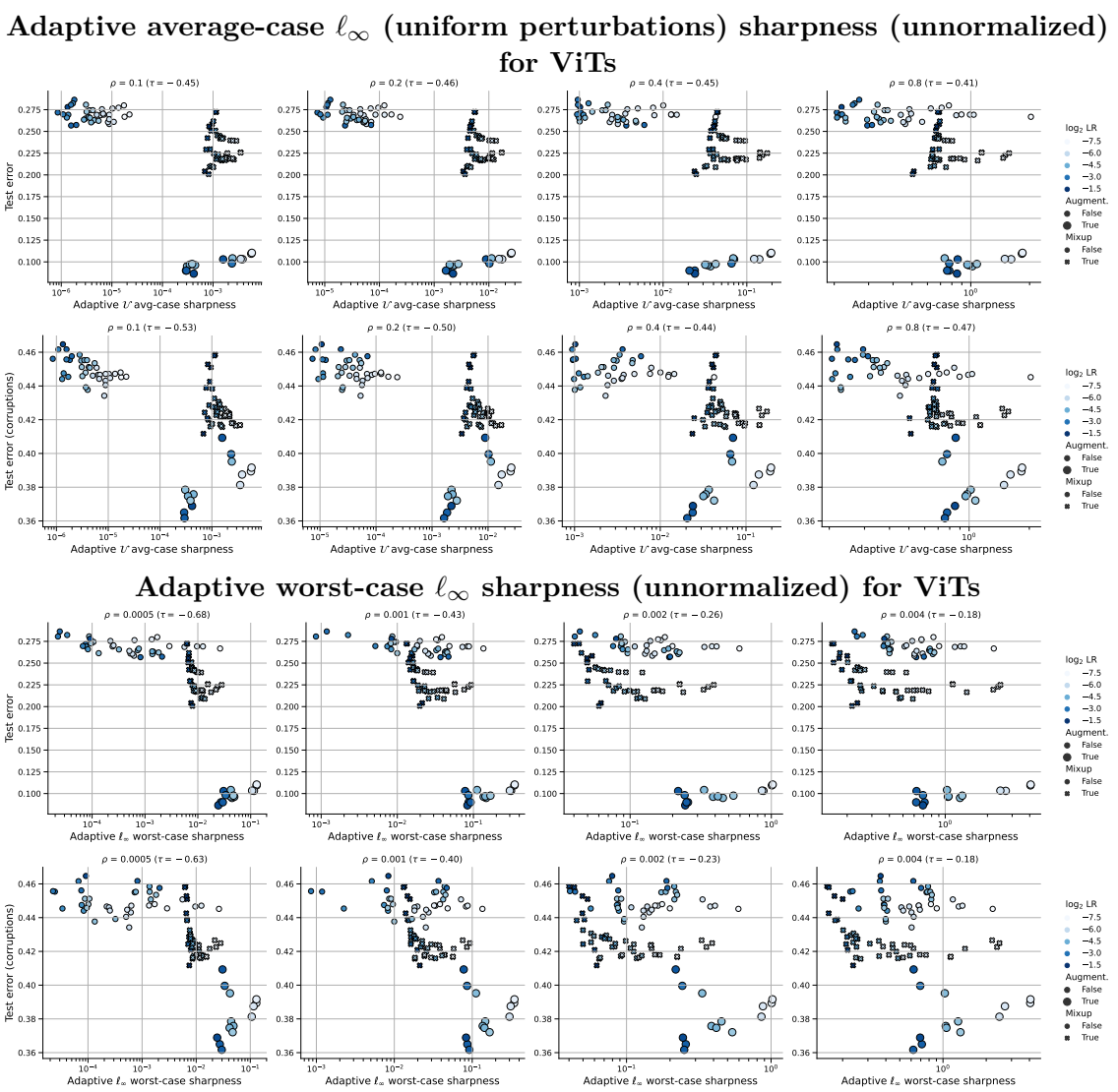


Figure 7.43: Average and worst-case ℓ_∞ adaptive sharpness definitions (unnormalized) and OOD test error (common corruptions) on CIFAR-10 for ViTs for different radii ρ.

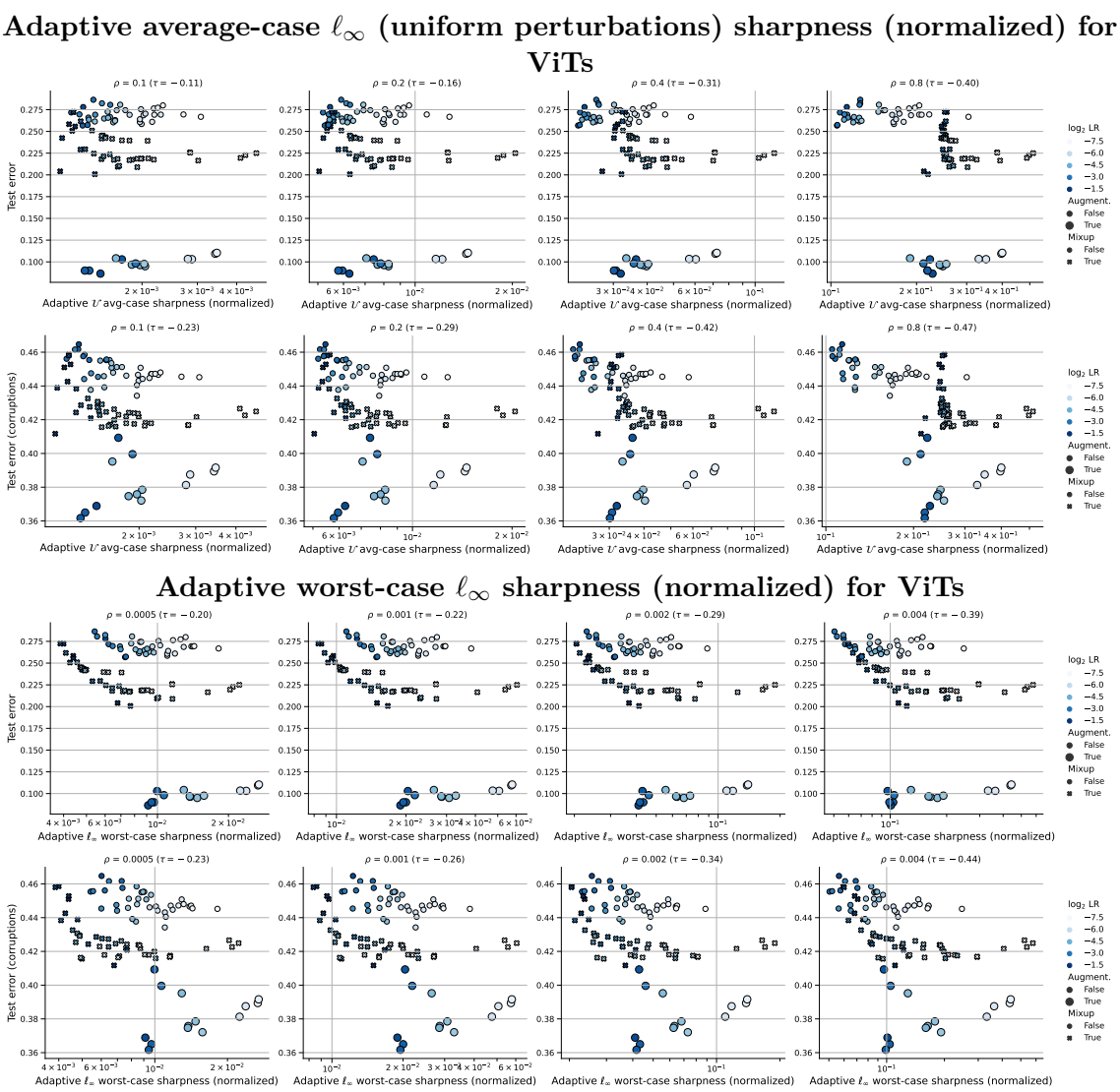


Figure 7.44: Average and worst-case ℓ_∞ adaptive sharpness definitions (normalized) vs. test error and OOD test error (common corruptions) on CIFAR-10 for ViTs for different radii ρ.

Conclusions

In this thesis, we have focused almost entirely on the setting where we train new task-specific models *from scratch*. However, during last few years, *foundation models* such as large language models (LLMs) (Bubeck et al., 2023) and vision-language models (VLMs) (Yang et al., 2023) have demonstrated exceptional capabilities in a wide range of natural language processing and computer vision tasks. Despite this, they are still often producing factually incorrect outputs (McKenna et al., 2023), tend to be overconfident (OpenAI, 2023), and are very susceptible to adversarial perturbations (Schlarmann and Hein, 2023), which undermines their reliability and trustworthiness. In particular, all efforts invested in improving the alignment and built-in safety filters of these models can be bypassed by transferable adversarial examples (Zou et al., 2023) or even manually crafted prompts (Wei et al., 2023). These weaknesses of generative models pose significant societal risks, encompassing threats such as cyber warfare and the proliferation of disinformation campaigns. These concerns have garnered recognition at both national and international levels, with an increasing number of regulations now mandating a prescribed level of transparency. For instance, the EU AI Act highlights the imperative of designing models to proactively prevent the generation of illegal content,[1] underscoring the urgency of addressing these challenges. By leveraging lessons from the works presented in this thesis, as future work, we plan to address these critical weaknesses in modern generative models by deepening our understanding of their generalization, more thorough automated evaluation, and enhancing their robustness.

Direction 1: Understanding the detrimental effects of the alignment training stage

The alignment phase is absolutely necessary to turn an LLM into a helpful assistant that supports dialogue-based interaction. Except making the LLM follow user's intent, it is also crucial to add a safety filter that refuses to provide harmful or hateful content to the user. However, the growing evidence suggests that this approach is detrimental at least to some capabilities present in the base model. As illustrated in the GPT-4 technical report (OpenAI, 2023), the alignment phase can significantly degrade calibration which is nearly perfect for the base model. This suggests that the model loses important capabilities,

[1] https://www.europarl.europa.eu/news/en/headlines/society/20230601STO93804/eu-ai-act-first-regulation-on-artificial-intelligence

such as judging about its own uncertainty, during the alignment phase. This can be seen as an additional evidence of implicit overoptimization or overfitting to the reward signal (Gao et al., 2023). Some of the deficiencies of aligned generative models such as factually incorrect outputs and overconfidence can partially stem from the alignment phase.

Direction 2: Developing automated safety evaluation methods

A precisely defined threat model, identifying the goals of potential attackers and the tools they may employ in each scenario (LLMs, VLMs, etc.), is essential for assessing model's robustness and safety. Without a clear metric to optimize, it becomes impossible to compare the performance of various systems and the effectiveness of attack algorithms. Once the threat models of interest are defined, we can focus on developing strong and efficient algorithms for adversarial perturbations. These attacks play a crucial role in benchmarking the field's progress towards safer systems. The absence of a well-defined threat model for text inputs stands in contrast to the relatively clear guidelines existing for image domains (Croce et al., 2021). In the context of text inputs, there is currently no established method for determining which perturbations, such as adding, removing, or modifying characters or words, are permissible to attackers. Furthermore, it remains unclear whether two different inputs should produce the same output under certain conditions. Moreover, the discrete nature of text input spaces complicates the application of gradient-based optimization, a common strategy in popular attacks. However, this inherent limitation may also be leveraged as an advantage, given the smaller attack space compared to vision-based tasks.

Direction 3: Developing trustworthy generative models

The overarching objective is to enhance the safety and robustness of foundation models, preventing their misuse in generating harmful content, while also ideally improving their interpretability. Adversarial training has emerged as the primary approach for enhancing the robustness of image classifiers against adversarial perturbations (Madry et al., 2018). This methodology can be readily extended to systems that incorporate an image encoder, such as CLIP (Radford et al., 2021). Moreover, training models with ℓ_p-bounded perturbations not only strengthens their resilience to these attacks but also imparts desirable side-effects, including better interpretability and generative capabilities (Tsipras et al., 2019). Thus, adversarial training is a natural candidate for improving the robustness and reliability of modern LLMs and VLMs. A crucial prerequisite is the adaptation of its framework for text inputs. In particular, an efficient algorithm for generating adversarial inputs at training time is needed. Initial attempts in this direction have proven unsuccessful (Jain et al., 2023), necessitating exploration of alternative defense strategies.

Outlook

If we extrapolate the remarkable progress achieved in the machine learning field during the last few years, the next years should showcase even more impressive capabilities of large models. LLM-powered agents are already capable of calling external functions suitable to the context, and this capability is expected to improve rapidly in the next years. Even with the current generative models such as GPT-4, significant improvements can be unlocked by incorporating external logic and orchestrating LLM calls in a more effective way. In light of these remarkable developments, it is very important to ensure safety and robustness of these systems for deployment in the real world. Since we know how to adversarially manipulate all kinds of machine learning models, including LLMs, manipulations of autonomous agents operating in the wild and being responsible for sensitive decisions will have a high incentive. In addition, it is also important to make sure that we understand sufficiently well the effect of learning algorithms and data used by them on generalization of these models and capabilities that they unlock. In summary, we need to make research progress in all these directions to ensure that these technologies are ready for responsible and safe deployment.

Bibliography

Linara Adilova, Maksym Andriushchenko, Michael Kamp, Asja Fischer, and Martin Jaggi. Layerwise linear mode connectivity. In *ICLR*, 2024.

Naman Agarwal, Surbhi Goel, and Cyril Zhang. Acceleration via fractal learning rate schedules. In *International Conference on Machine Learning*, pages 87–99. PMLR, 2021.

N. Akhtar and A. Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access*, 6:14410–14430, 2018.

A. Al-Dujaili and U.-M. O'Reilly. There are no bit parts for sign bits in black-box attacks. In *ICLR*, 2020.

Jean-Baptiste Alayrac, Jonathan Uesato, Po-Sen Huang, Robert Stanforth, Alhussein Fawzi, and Pushmeet Kohli. Are labels required for improving adversarial robustness? *NeurIPS*, 2019.

Motasem Alfarra, Juan C. Perez, Adel Bibi, Ali Thabet, Pablo Arbelaez, and Bernard Ghanem. Clustr: Clustering training for robustness. *arXiv preprint arXiv:2006.07682*, 2020.

M. Alzantot, Y. Sharma, S. Chakraborty, and M. Srivastava. Genattack: practical black-box attacks with gradient-free optimization. In *Genetic and Evolutionary Computation Conference (GECCO)*, 2019.

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. Generating natural language adversarial examples. In *EMNLP*, 2018.

Maksym Andriushchenko. Adversarial attacks on gpt-4 via simple random search, 2023. URL https://www.andriushchenko.me/gpt4adv.pdf.

Maksym Andriushchenko and Nicolas Flammarion. Understanding and improving fast adversarial training. In *NeurIPS*, 2020.

Maksym Andriushchenko and Nicolas Flammarion. Towards understanding sharpness-aware minimization. In *ICML*, 2022.

Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion, and Matthias Hein. Square attack: a query-efficient black-box adversarial attack via random search. In *ECCV*, 2020.

Maksym Andriushchenko, Xiaoyang Rebecca Li, Geoffrey Oxholm, Thomas Gittings, Tu Bui, Nicolas Flammarion, and John Collomosse. Aria: Adversarially robust image attribution for content provenance. In *CVPR Workshop on Media Forensics*, 2022.

Maksym Andriushchenko, Dara Bahri, Hossein Mobahi, and Nicolas Flammarion. Sharpness-aware minimization leads to low-rank features. In *NeurIPS*, 2023a.

Maksym Andriushchenko, Francesco Croce, Maximilian Müller, Matthias Hein, and Nicolas Flammarion. A modern look at the relationship between sharpness and generalization. In *ICML*, 2023b.

Maksym Andriushchenko, Francesco D'Angelo, Aditya Varre, and Nicolas Flammarion. Why do we need weight decay in modern deep learning? *NeurIPS 2023 Workshop on Mathematics of Modern Machine Learning*, 2023c.

Maksym Andriushchenko, Aditya Varre, Loucas Pillaud-Vivien, and Nicolas Flammarion. SGD with large step sizes learns sparse features. In *ICML*, 2023d.

Sanjeev Arora, Zhiyuan Li, and Abhishek Panigrahi. Understanding gradient descent on edge of stability in deep learning. *ICML*, 2022.

Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *ICML*, 2018.

Matan Atzmon, Niv Haim, Lior Yariv, Ofer Israelov, Haggai Maron, and Yaron Lipman. Controlling neural level sets. *NeurIPS*, 2019.

Maximilian Augustin, Alexander Meinke, and Matthias Hein. Adversarial robustness on in- and out-distribution improves explainability. *ECCV*, 2020.

Andrei Barbu, David Mayo, Julian Alverio, William Luo, Christopher Wang, Dan Gutfreund, Josh Tenenbaum, and Boris Katz. Objectnet: A large-scale bias-controlled dataset for pushing the limits of object recognition models. In *NeurIPS*, 2019.

O. Bastani, Y. Ioannou, L. Lampropoulos, D. Vytiniotis, A. Nori, and A. Criminisi. Measuring neural net robustness with constraints. In *NeurIPS*, 2016.

Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovski. *Robust optimization*. Princeton Series in Applied Mathematics. Princeton University Press, Princeton, NJ, 2009.

Philipp Benz, Chaoning Zhang, Adil Karjauv, and In So Kweon. Robustness may be at odds with fairness: An empirical study on class-wise accuracy. *arXiv preprint arXiv:2010.13365*, 2020.

Gaspard Beugnot, Julien Mairal, and Alessandro Rudi. On the benefits of large learning rates for kernel methods. *arXiv preprint arXiv:2202.13733*, 2022.

A. N. Bhagoji, W. He, B. Li, and D. Song. Practical black-box attacks on deep neural networks using efficient query mechanisms. In *ECCV*, 2018.

B. Biggio and F. Roli. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84:317–331, 2018a.

Battista Biggio and Fabio Roli. Wild patterns: ten years after the rise of adversarial machine learning. *Pattern Recognition*, 2018b.

Devansh Bisla, Jing Wang, and Anna Choromanska. Low-pass filtering sgd for recovering flat optima in the deep learning optimization landscape. *AISTATS*, 2022.

Nils Bjorck, Carla P Gomes, Bart Selman, and Kilian Q Weinberger. Understanding batch normalization. *Advances in neural information processing systems*, 31, 2018.

Guy Blanc, Neha Gupta, Gregory Valiant, and Paul Valiant. Implicit regularization for deep neural networks driven by an ornstein-uhlenbeck like process. In *Conference on learning theory*, pages 483–513. PMLR, 2020.

Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint aXiv:2004.10934*, 2020.

Etienne Boursier, Loucas Pillaud-Vivien, and Nicolas Flammarion. Gradient flow dynamics of shallow relu networks for square loss and orthogonal inputs. *arXiv preprint arXiv:2206.00939*, 2022.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, 2004.

W. Brendel, J. Rauber, and M. Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. In *ICLR*, 2018a.

Wieland Brendel, Jonas Rauber, Alexey Kurakin, Nicolas Papernot, Behar Veliqi, Marcel Salathé, Sharada P Mohanty, and Matthias Bethge. Adversarial vision challenge. In *NeurIPS Competition Track*, 2018b.

T. B. Brown, D. Mané, A. Roy, M. Abadi, and J. Gilmer. Adversarial patch. In *NeurIPS 2017 Workshop on Machine Learning and Computer Security*, 2017.

Tom B Brown, Nicholas Carlini, Chiyuan Zhang, Catherine Olsson, Paul Christiano, and Ian Goodfellow. Unrestricted adversarial examples. *arXiv preprint arXiv:1809.08352*, 2018.

T. Brunner, F. Diehl, M. T. Le, and A. Knoll. Guessing smart: biased sampling for efficient black-box adversarial attacks. In *ICCV*, 2019.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.

Jacob Buckman, Aurko Roy, Colin Raffel, and Ian Goodfellow. Thermometer encoding: One hot way to resist adversarial examples. In *ICLR*, 2018.

Dan A. Calian, Florian Stimberg, Olivia Wiles, Sylvestre-Alvise Rebuffi, Andras Gyorgy, Timothy Mann, and Sven Gowal. Defending against image corruptions through adversarial augmentations. *arXiv*, 2021.

N. Carlini and D. Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *ACM Workshop on Artificial Intelligence and Security*, 2017.

Nicholas Carlini. A critique of the deepsec platform for security analysis of deep learning models. *arXiv preprint arXiv:1905.07112*, 2019.

Nicholas Carlini. A complete list of all (arxiv) adversarial example papers. https://nicholas.carlini.com/writing/2019/all-adversarial-example-papers.html, 2021. Accessed: 2021-06-08.

Nicholas Carlini, Guy Katz, Clark Barrett, and David L Dill. Provably minimally-distorted adversarial examples. *arXiv preprint arXiv:1709.10207*, 2017.

Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian Goodfellow, Aleksander Madry, and Alexey Kurakin. On evaluating adversarial robustness. *arXiv preprint arXiv:1902.06705*, 2019a.

Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 267–284, 2019b.

Yair Carmon, Aditi Raghunathan, Ludwig Schmidt, John C Duchi, and Percy S Liang. Unlabeled data improves adversarial robustness. In *NeurIPS*, 2019.

Junbum Cha, Sanghyuk Chun, Kyungjae Lee, Han-Cheol Cho, Seunghyun Park, Yunsung Lee, and Sungrae Park. Swad: Domain generalization by seeking flat minima. *NeurIPS*, 34:22405–22418, 2021.

Alvin Chan, Yi Tay, Yew Soon Ong, and Jie Fu. Jacobian adversarially regularized networks for robustness. *ICLR*, 2020.

Pratik Chaudhari, Anna Choromanska, Stefano Soatto, Yann LeCun, Carlo Baldassi, Christian Borgs, Jennifer Chayes, Levent Sagun, and Riccardo Zecchina. Entropy-SGD: Biasing gradient descent into wide valleys. *Journal of Statistical Mechanics: Theory and Experiment*, 2019(12):124018, 2016.

J. Chen, M. I Jordan, and Wainwright M. J. HopSkipJumpAttack: a query-efficient decision-based attack. arXiv preprint arXiv:1904.02144, 2019.

Jinghui Chen and Quanquan Gu. Rays: A ray searching method for hard-label adversarial attack. In *KDD*, 2020.

Jinghui Chen, Yu Cheng, Zhe Gan, Quanquan Gu, and Jingjing Liu. Efficient robust training via backward smoothing. *arXiv*, 2020a.

Lei Chen and Joan Bruna. On gradient descent convergence beyond the edge of stability. *arXiv preprint arXiv:2206.04172*, 2022.

P. Chen, Y. Sharma, H. Zhang, J. Yi, and C. Hsieh. Ead: Elastic-net attacks to deep neural networks via adversarial examples. In *AAAI*, 2018.

Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *10th ACM Workshop on Artificial Intelligence and Security - AISec '17*. ACM Press, 2017. ISBN 9781450352024.

Tianlong Chen, Sijia Liu, Shiyu Chang, Yu Cheng, Lisa Amini, and Zhangyang Wang. Adversarial robustness: From self-supervised pre-training to fine-tuning. In *CVPR*, 2020b.

Xiangning Chen, Cho-Jui Hsieh, and Boqing Gong. When vision transformers outperform resnets without pre-training or strong data augmentations? *ICLR*, 2022.

M. Cheng, T. Le, P.-Y. Chen, J. Yi, H. Zhang, and C.-J. Hsieh. Query-efficient hard-label black-box attack: An optimization-based approach. In *ICLR*, 2019a.

S. Cheng, Y. Dong, T. Pang, H. Su, and J. Zhu. Improving black-box adversarial attacks with a transfer-based prior. In *NeurIPS*, 2019b.

Lenaic Chizat and Francis Bach. Implicit bias of gradient descent for wide two-layer neural networks trained with the logistic loss. In *Conference on Learning Theory*, pages 1305–1338. PMLR, 2020a.

Lénaïc Chizat and Francis Bach. Implicit bias of gradient descent for wide two-layer neural networks trained with the logistic loss. In *COLT*, 2020b.

Lenaic Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. *Advances in Neural Information Processing Systems*, 32, 2019.

Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *CVPR*, 2014.

Jeremy M Cohen, Elan Rosenfeld, and J Zico Kolter. Certified adversarial robustness via randomized smoothing. *ICML*, 2019.

Jeremy M Cohen, Simran Kaur, Yuanzhi Li, J Zico Kolter, and Ameet Talwalkar. Gradient descent on neural networks typically occurs at the edge of stability. *arXiv preprint arXiv:2103.00065*, 2021.

F. Croce and M. Hein. Sparse and imperceivable adversarial attacks. In *ICCV*, 2019.

F. Croce and M. Hein. Minimally distorted adversarial examples with a fast adaptive boundary attack. In *ICML*, 2020a.

Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. *ICML*, 2020b.

Francesco Croce, Maksym Andriushchenko, Naman D Singh, Nicolas Flammarion, and Matthias Hein. Sparse-rs: a versatile framework for query-efficient sparse black-box adversarial attacks. In *ECCV Workshop on Adversarial Robustness in the Real World*, 2020.

Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Edoardo Debenedetti, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. Robustbench: a standardized adversarial robustness benchmark. *NeurIPS Datasets and Benchmarks Track*, 2021.

Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. *NeurIPS*, 2020.

Jiequan Cui, Shu Liu, Liwei Wang, and Jiaya Jia. Learnable boundary guided adversarial training. *ICCV*, 2021.

Alex Damian, Tengyu Ma, and Jason D Lee. Label noise sgd provably prefers flat global minimizers. *Advances in Neural Information Processing Systems*, 34:27449–27461, 2021.

Alex Damian, Eshaan Nichani, and Jason D Lee. Self-stabilization: The implicit bias of gradient descent at the edge of stability. *ICLR*, 2023.

Luke N Darlow, Elliot J Crowley, Antreas Antoniou, and Amos J Storkey. Cinic-10 is not imagenet or cifar-10. *arXiv preprint arXiv:1810.03505*, 2018.

D. Davis and D. Drusvyatskiy. Stochastic model-based minimization of weakly convex functions. *SIAM Journal on Optimization*, 29(1):207–239, 2019.

Edoardo Debenedetti, Zishen Wan, Maksym Andriushchenko, Vikash Sehwag, Kshitij Bhardwaj, and Bhavya Kailkhura. Scaling compute is not all you need for adversarial robustness. *ICLR 2024 Workshop on Reliable and Responsible Foundation Models*, 2023.

Jonas Degrave, Federico Felici, Jonas Buchli, Michael Neunert, Brendan Tracey, Francesco Carpanese, Timo Ewalds, Roland Hafner, Abbas Abdolmaleki, Diego de Las Casas, et al. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602 (7897):414–419, 2022.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

Emily L Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. *Advances in neural information processing systems*, 27, 2014.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019.

Gavin Weiguang Ding, Luyu Wang, and Xiaomeng Jin. AdverTorch v0.1: An adversarial robustness toolbox based on pytorch. *arXiv preprint arXiv:1902.07623*, 2019.

Gavin Weiguang Ding, Yash Sharma, Kry Yik Chau Lui, and Ruitong Huang. Mma training: Direct input space margin maximization through adversarial training. In *ICLR*, 2020.

Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. In *ICML*, pages 1019–1028. PMLR, 2017.

Yinpeng Dong, Qi-An Fu, Xiao Yang, Tianyu Pang, Hang Su, Zihao Xiao, and Jun Zhu. Benchmarking adversarial robustness on image classification. In *CVPR*, 2020.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021.

J. Du, H. Zhang, J. T. Zhou, Y. Yang, and J. Feng. Query-efficient meta attack to deep neural networks. In *ICLR*, 2020.

Jiawei Du, Zhou Daquan, Jiashi Feng, Vincent Tan, and Joey Tianyi Zhou. Sharpness-aware training for free. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=xK6wRfL2mv7.

J. Duchi, M. Jordan, M. Wainwright, and A. Wibisono. Optimal rates for zero-order convex optimization: The power of two function evaluations. *IEEE Transactions on Information Theory*, 61(5):2788–2806, 2015.

Gintare Karolina Dziugaite and Daniel Roy. Entropy-sgd optimizes the prior of a pac-bayes bound: Generalization properties of entropy-sgd and data-dependent priors. In *ICML*, 2018.

Gintare Karolina Dziugaite, Alexandre Drouin, Brady Neal, Nitarshan Rajkumar, Ethan Caballero, Linbo Wang, Ioannis Mitliagkas, and Daniel M Roy. In search of robust measures of generalization. *arXiv preprint arXiv:2010.11924*, 2020.

Logan Engstrom, Andrew Ilyas, and Anish Athalye. Evaluating and understanding the robustness of adversarial logit pairing. *NeurIPS 2018 Workshop on Security in Machine Learning*, 2018.

Logan Engstrom, Andrew Ilyas, Hadi Salman, Shibani Santurkar, and Dimitris Tsipras. Robustness (python library), 2019a. URL https://github.com/MadryLab/robustness.

Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Brandon Tran, and Aleksander Madry. Adversarial robustness as a prior for learned representations. *arXiv preprint arXiv:1906.00945*, 2019b.

Logan Engstrom, Brandon Tran, Dimitris Tsipras, Ludwig Schmidt, and Aleksander Madry. Exploring the landscape of spatial robustness. In *ICML*, 2019c.

Andre Esteva, Alexandre Robicquet, Bharath Ramsundar, Volodymyr Kuleshov, Mark DePristo, Katherine Chou, Claire Cui, Greg Corrado, Sebastian Thrun, and Jeff Dean. A guide to deep learning in healthcare. *Nature medicine*, 25(1):24–29, 2019.

Christian Etmann. A closer look at double backpropagation. *arXiv preprint ArXiv:1906.06637*, 2019.

Ivan Evtimov, Kevin Eykholt, Earlence Fernandes, Tadayoshi Kohno, Bo Li, Atul Prakash, Amir Rahmati, and Dawn Song. Robust physical-world attacks on machine learning models. In *CVPR*, 2018.

Fartash Faghri, Cristina Vasconcelos, David J Fleet, Fabian Pedregosa, and Nicolas Le Roux. Bridging the gap between adversarial robustness and optimization bias. *arXiv preprint arXiv:2102.08868*, 2021.

Alhussein Fawzi and Pascal Frossard. Manitest: Are classifiers really invariant? In *BMVC*, 2015.

Alhussein Fawzi and Pascal Frossard. Measuring the effect of nuisance variables on classifiers. In *British Machine Vision Conference (BMVC)*, 2016.

Nicolas Ford, Justin Gilmer, Nicolas Carlini, and Dogus Cubuk. Adversarial examples are a natural consequence of test error in noise. In *ICML*, 2019.

Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. In *ICLR*, 2021.

Stanislav Fort, Andrew Brock, Razvan Pascanu, Soham De, and Samuel L Smith. Drawing multiple augmentation samples per image during training efficiently decreases test error. *arXiv preprint arXiv:2105.13343*, 2021.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning*, pages 10835–10866. PMLR, 2023.

Jonas Geiping, Micah Goldblum, Phillip E Pope, Michael Moeller, and Tom Goldstein. Stochastic training is not necessary for generalization. In *International Conference on Learning Representations*, 2022.

Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A. Wichmann, and Wieland Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. *ICLR*, 2019.

Saeed Ghadimi and Guanghui Lan. Stochastic first- and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368, 2013.

Justin Gilmer, Ryan P Adams, Ian Goodfellow, David Andersen, and George E Dahl. Motivating the rules of the game for adversarial example research. *arXiv preprint arXiv:1807.06732*, 2018.

Amir Globerson and Sam Roweis. Nightmare at test time: robust learning by feature deletion. *ICML*, 2006.

Rafael C Gonzales and Richard E Woods. *Digital image processing (2nd edition)*. Prentice Hall New Jersey, 2002.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *ICLR*, 2015.

Dou Goodman, Hao Xin, Wang Yang, Wu Yuesheng, Xiong Junfeng, and Zhang Huan. Advbox: a toolbox to generate adversarial examples that fool neural networks. *arXiv preprint arXiv:2001.05574*, 2020.

Sven Gowal, Krishnamurthy (Dj) Dvijotham, Robert Stanforth, Rudy Bunel, Chongli Qin, Jonathan Uesato, Relja Arandjelovic, Timothy Mann, and Pushmeet Kohli. Scalable verified training for provably robust image classification. In *ICCV*, 2019a.

Sven Gowal, Jonathan Uesato, Chongli Qin, Po-Sen Huang, Timothy Mann, and Pushmeet Kohli. An alternative surrogate loss for pgd-based adversarial testing. *arXiv preprint arXiv:1910.09338*, 2019b.

Sven Gowal, Chongli Qin, Jonathan Uesato, Timothy Mann, and Pushmeet Kohli. Uncovering the limits of adversarial training against norm-bounded adversarial examples. *arXiv*, 2020.

Robert Mansel Gower, Nicolas Loizou, Xun Qian, Alibek Sailanbayev, Egor Shulgin, and Peter Richtárik. SGD: General analysis and improved rates. In *ICML*, 2019.

Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

Diego Granziol. Flatness is a false friend. *arXiv preprint arXiv:2006.09091*, 2020.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE ICASSP*, 2013.

Kathrin Grosse, Nicolas Papernot, Praveen Manoharan, Michael Backes, and Patrick McDaniel. Adversarial examples for malware detection. In *European symposium on research in computer security*, 2017.

S. Gu and L. Rigazio. Towards deep neural network architectures robust to adversarial examples. In *ICLR Workshop*, 2015a.

Shixiang Gu and Luca Rigazio. Towards deep neural network architectures robust to adversarial examples. *ICLR workshops*, 2015b.

C. Guo, J. S Frank, and K. Q Weinberger. Low frequency adversarial perturbation. In *UAI*, 2019a.

C. Guo, J. R Gardner, Y. You, A. G. Wilson, and K. Q. Weinberger. Simple black-box adversarial attacks. In *ICML*, 2019b.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, pages 1321–1330. PMLR, 2017.

Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens van der Maaten. Countering adversarial images using input transformations. In *ICLR*, 2018.

U. Haagerup. The best constants in the Khintchine inequality. *Studia Math.*, 70(3): 231–283, 1981.

Jeff Z HaoChen, Colin Wei, Jason D Lee, and Tengyu Ma. Shape matters: Understanding the implicit bias of the noise covariance. *arXiv preprint arXiv:2006.08680*, 2020.

Kosuke Haruki, Taiji Suzuki, Yohei Hamakawa, Takeshi Toda, Ryuji Sakai, Masahiro Ozawa, and Mitsuhiro Kimura. Gradient noise convolution (gnc): Smoothing loss function for distributed large-batch sgd. *arXiv preprint arXiv:1906.10822*, 2019.

Haowei He, Gao Huang, and Yang Yuan. Asymmetric valleys: Beyond sharp and flat local minima. In *NeurIPS*, 2019.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *ICCV*, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016a.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. *ECCV*, 2016b.

M. Hein and M. Andriushchenko. Formal guarantees on the robustness of a classifier against adversarial manipulation. In *NeurIPS*, 2017a.

Matthias Hein and Maksym Andriushchenko. Formal guarantees on the robustness of a classifier against adversarial manipulation. *NeurIPS*, 2017b.

Matthias Hein, Maksym Andriushchenko, and Julian Bitterwolf. Why relu networks yield high-confidence predictions far away from the training data and how to mitigate the problem. In *CVPR*, 2019.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *ICLR*, 2019.

Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *ICLR*, 2017.

Dan Hendrycks, Kimin Lee, and Mantas Mazeika. Using pre-training can improve model robustness and uncertainty. In *ICML*, 2019.

Dan Hendrycks, Norman Mu, Ekin D. Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. Augmix: A simple data processing method to improve robustness and uncertainty. *ICLR*, 2020.

Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, Dawn Song, Jacob Steinhardt, and Justin Gilmer. The many faces of robustness: A critical analysis of out-of-distribution generalization. *ICCV*, 2021a.

Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, Dawn Song, Jacob Steinhardt, and Justin Gilmer. The many faces of robustness: A critical analysis of out-of-distribution generalization. *ICCV*, 2021b.

Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. *CVPR*, 2021c.

Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015.

Sepp Hochreiter and Jürgen Schmidhuber. Simplifying neural nets by discovering flat minima. In *NeurIPS*, 1995.

Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *JMLR*, 22(241):1–124, 2021.

Elad Hoffer, Itay Hubara, and Daniel Soudry. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. In *NeurIPS*, 2017.

J Edward Hu, Adith Swaminathan, Hadi Salman, and Greg Yang. Improved image wasserstein attacks and defenses. *ICLR Workshop: Towards Trustworthy ML: Rethinking Security and Privacy for ML*, 2020.

Lang Huang, Chao Zhang, and Hongyang Zhang. Self-adaptive training: beyond empirical risk minimization. *NeurIPS*, 2020.

Peter J. Huber. *Robust statistics*. John Wiley & Sons, Inc., New York, 1981.

A. Ilyas, L. Engstrom, A. Athalye, and J. Lin. Black-box adversarial attacks with limited queries and information. In *ICML*, 2018.

A. Ilyas, L. Engstrom, and A. Madry. Prior convictions: Black-box adversarial attacks with bandits and priors. In *ICLR*, 2019a.

A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran, and A. Madry. Adversarial examples are not bugs, they are features. *NeurIPS*, 2019b.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. In *UAI*, 2018.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

Arthur Jacot, François Ged, Berfin Şimşek, Clément Hongler, and Franck Gabriel. Saddle-to-saddle dynamics in deep linear networks: Small initialization training, symmetry, and sparsity. In *International Conference on Machine Learning*, 2021.

Neel Jain, Avi Schwarzschild, Yuxin Wen, Gowthami Somepalli, John Kirchenbauer, Ping-yeh Chiang, Micah Goldblum, Aniruddha Saha, Jonas Geiping, and Tom Goldstein. Baseline defenses for adversarial attacks against aligned language models. *arXiv preprint arXiv:2309.00614*, 2023.

Yunseok Jang, Tianchen Zhao, Seunghoon Hong, and Honglak Lee. Adversarial defense via learning to generate diverse attacks. *ICCV*, 2019.

Stanislaw Jastrzebski, Zachary Kenton, Devansh Arpit, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amos Storkey. Three factors influencing minima in sgd. *arXiv preprint arXiv:1711.04623*, 2017.

Stanislaw Jastrzebski, Devansh Arpit, Oliver Astrand, Giancarlo B Kerg, Huan Wang, Caiming Xiong, Richard Socher, Kyunghyun Cho, and Krzysztof J Geras. Catastrophic fisher explosion: Early phase fisher matrix impacts generalization. In *Proceedings of the 38th International Conference on Machine Learning*. PMLR, 2021.

Yiding Jiang, Behnam Neyshabur, Hossein Mobahi, Dilip Krishnan, and Samy Bengio. Fantastic generalization measures and where to find them. In *ICLR*, 2019.

Kam-Chuen Jim, C Lee Giles, and Bill G Horne. An analysis of noise in recurrent neural networks: convergence and generalization. In *IEEE Transactions on Neural Networks*, 1996.

Charles Jin and Martin Rinard. Manifold regularization for adversarial robustness. *arXiv*, 2020.

An Ju. Generative models as a robust alternative for image classification: Progress and challenges. *PhD thesis, UC Berkeley*, 2021.

Anatoli Juditsky, Arkadi Nemirovski, and Claire Tauvel. Solving variational inequalities with stochastic mirror-prox algorithm. *Stochastic Systems*, 2011.

Daniel Kang, Yi Sun, Tom Brown, Dan Hendrycks, and Jacob Steinhardt. Transfer of adversarial robustness between perturbation types. *arXiv preprint arXiv:1905.01034*, 2019a.

Daniel Kang, Yi Sun, Dan Hendrycks, Tom Brown, and Jacob Steinhardt. Testing robustness against unforeseen adversaries. *arXiv preprint arXiv:1908.08016*, 2019b.

H. Kannan, A. Kurakin, and I. Goodfellow. Adversarial logit pairing. arXiv preprint arXiv:1803.06373, 2018.

Hamed Karimi, Julie Nutini, and Mark Schmidt. Linear convergence of gradient and proximal-gradient methods under the Polyak-Lojasiewicz condition. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2016.

Danny Karmon, Daniel Zoran, and Yoav Goldberg. Lavan: Localized and visible adversarial noise. In *ICML*, 2018.

Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. Reluplex: an efficient smt solver for verifying deep neural networks. *ICCAV*, 2017.

Simran Kaur, Jeremy Cohen, and Zachary C Lipton. Are perceptually-aligned gradients a general property of robust classifiers? In *NeurIPS Workshop: Science Meets Engineering of Deep Learning*, 2019.

Simran Kaur, Jeremy Cohen, and Zachary C Lipton. On the maximum hessian eigenvalue and generalization. *arXiv preprint arXiv:2206.10654*, 2022.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *ICLR*, 2016.

Jungeum Kim and Xiao Wang. Sensible adversarial learning. *OpenReview*, 2019.

Klim Kireev, Maksym Andriushchenko, and Nicolas Flammarion. On the effectiveness of adversarial training against common corruptions. *arXiv*, 2021.

Klim Kireev, Maksym Andriushchenko, Carmela Troncoso, and Nicolas Flammarion. Transferable adversarial robustness for categorical data via universal robust embeddings. In *NeurIPS*, 2023.

Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, et al. Wilds: A benchmark of in-the-wild distribution shifts. *arXiv preprint arXiv:2012.07421*, 2020.

Galina Korpelevich. Extragradient method for finding saddle points and other problems. In *Matekon*, 1977.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *Technical Report*, 2009.

Bogdan Kulynych, Rebekah Overdorf, Carmela Troncoso, and Seda Gürses. Pots: protective optimization technologies. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pages 177–188, 2020.

Souvik Kundu, Mahdi Nazemi, Peter A Beerel, and Massoud Pedram. A tunable robust pruning framework through dynamic network rewiring of dnns. *ASP-DAC*, 2021.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *ICLR Workshop*, 2017. URL https://openreview.net/forum?id=HJGU3Rodl.

Alexey Kurakin, Ian Goodfellow, Samy Bengio, Yinpeng Dong, Fangzhou Liao, Ming Liang, Tianyu Pang, Jun Zhu, Xiaolin Hu, Cihang Xie, et al. Adversarial attacks and defences competition. In *NeurIPS Competition Track*, 2018.

Jungmin Kwon, Jeongseop Kim, Hyunseo Park, and In Kwon Choi. ASAM: Adaptive sharpness-aware minimization for scale-invariant learning of deep neural networks. In *ICML*, 2021.

Cassidy Laidlaw and Soheil Feizi. Functional adversarial attacks. In *NeurIPS*, 2019.

Cassidy Laidlaw, Sahil Singla, and Soheil Feizi. Perceptual adversarial robustness: Defense against unseen threat models. *arXiv preprint arXiv:2006.12655*, 2020.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553), 2015.

Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 2012.

Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. Certified robustness to adversarial examples with differential privacy. In *2019 IEEE S&P*, 2019.

Aitor Lewkowycz, Yasaman Bahri, Ethan Dyer, Jascha Sohl-Dickstein, and Guy Gur-Ari. The large learning rate phase of deep learning: the catapult mechanism. *arXiv preprint arXiv:2003.02218*, 2020.

Linyi Li, Maurice Weber, Xiaojun Xu, Luka Rimanic, Bhavya Kailkhura, Tao Xie, Ce Zhang, and Bo Li. Tss: Transformation-specific smoothing for robustness certification. In *ACM CCS*, 2021.

Qianxiao Li, Cheng Tai, and E Weinan. Stochastic modified equations and dynamics of stochastic gradient algorithms i: Mathematical foundations. *The Journal of Machine Learning Research*, 20(1):1474–1520, 2019a.

Y. Li, L. Li, L. Wang, T. Zhang, and B. Gong. Nattack: Learning the distributions of adversarial examples for an improved black-box attack on deep neural networks. In *ICML*, 2019b.

Yaxin Li, Wei Jin, Han Xu, and Jiliang Tang. Deeprobust: A pytorch library for adversarial attacks and defenses. *arXiv preprint arXiv:2005.06149*, 2020a.

Yingzhen Li, John Bradshaw, and Yash Sharma. Are generative classifiers more robust to adversarial attacks? In *ICML*, 2019c.

Yuanzhi Li, Tengyu Ma, and Hongyang Zhang. Algorithmic regularization in over-parameterized matrix sensing and neural networks with quadratic activations. In *Conference On Learning Theory*, pages 2–47. PMLR, 2018.

Yuanzhi Li, Colin Wei, and Tengyu Ma. Towards explaining the regularization effect of initial large learning rate in training neural networks. In *Advances in Neural Information Processing Systems*, pages 11674–11685, 2019d.

Zhiyuan Li and Sanjeev Arora. An exponential learning rate schedule for deep learning. *arXiv preprint arXiv:1910.07454*, 2019.

Zhiyuan Li, Kaifeng Lyu, and Sanjeev Arora. Reconciling modern deep learning with traditional optimization analyses: The intrinsic learning rate. In *Advances in Neural Information Processing Systems*, volume 33, pages 14544–14555, 2020b.

Zhiyuan Li, Tianhao Wang, and Sanjeev Arora. What happens after sgd reaches zero loss?– a mathematical framework. In *International Conference on Learning Representations*, 2022.

Tengyuan Liang, Tomaso Poggio, Alexander Rakhlin, and James Stokes. Fisher-rao metric, geometry, and complexity of neural networks. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 888–896. PMLR, 2019.

Tao Lin, Lingjing Kong, Sebastian Stich, and Martin Jaggi. Extrapolation for large-batch training in deep learning. In *ICML*, 2020.

Y. Lin, H. Jiang, and H. Jiang. Bandlimiting neural networks against adversarial attacks. arXiv preprint arXiv:1905.12797, 2019.

Xiang Ling, Shouling Ji, Jiaxu Zou, Jiannan Wang, Chunming Wu, Bo Li, and Ting Wang. Deepsec: A uniform platform for security analysis of deep learning model. In *IEEE S&P*, 2019.

Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989.

Shengchao Liu, Dimitris Papailiopoulos, and Dimitris Achlioptas. Bad global minima exist and SGD can reach them. In *NeurIPS*, 2019.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Daniel Lowd and Christopher Meek. Adversarial learning. In *KDD*, 2005.

Kaifeng Lyu and Jian Li. Gradient descent maximizes the margin of homogeneous neural networks. In *International Conference on Learning Representations*, 2020.

Kaifeng Lyu, Zhiyuan Li, and Sanjeev Arora. Understanding the generalization benefit of normalization layers: Sharpness reduction. *NeurIPS*, 2022.

Chao Ma and Lexing Ying. The Sobolev regularization effect of stochastic gradient descent. *arXiv preprint arXiv:2105.13462*, 2021.

Chao Ma, Lei Wu, and Lexing Ying. The multiscale structure of neural network loss functions: The effect on optimization and origin. *arXiv preprint arXiv:2204.11326*, 2022.

A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2018.

Thibault Maho, Benoît Bonnet, Teddy Furon, and Erwan Le Merrer. Robic: A benchmark suite for assessing classifiers robustness. *arXiv preprint arXiv:2102.05368*, 2021.

Pratyush Maini, Eric Wong, and J Zico Kolter. Adversarial robustness against the union of multiple perturbation models. In *ICML*, 2020.

Chengzhi Mao, Ziyuan Zhong, Junfeng Yang, Carl Vondrick, and Baishakhi Ray. Metric learning for adversarial robustness. *NeurIPS*, 2019.

J. Matyas. Random optimization. *Automation and Remote control*, 26(2):246–253, 1965.

R. Thomas McCoy, Junghyun Min, and Tal Linzen. BERTs of a feather do not generalize together: Large variability in generalization across models with similar test set

performance. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, November 2020.

Tom McCoy, Ellie Pavlick, and Tal Linzen. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *ACL*, 2019.

Nick McKenna, Tianyi Li, Liang Cheng, Mohammad Javad Hosseini, Mark Johnson, and Mark Steedman. Sources of hallucination by large language models on inference tasks. *arXiv preprint arXiv:2305.14552*, 2023.

Marco Melis, Ambra Demontis, Maura Pintor, Angelo Sotgiu, and Battista Biggio. secml: A python library for secure and explainable machine learning. *arXiv preprint arXiv:1912.10013*, 2019.

L. Meunier, J. Atif, and O. Teytaud. Yet another but more efficient black-box adversarial attack: tiling and evolution strategies. arXiv preprint, arXiv:1910.02244, 2019.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. Mixed precision training. *ICLR*, 2018.

Apostolos Modas, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. Sparsefool: a few pixels make a big difference. In *CVPR*, 2019.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Jonathan Uesato, and Pascal Frossard. Robustness via curvature regularization, and vice versa. *CVPR*, 2019a.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Jonathan Uesato, and Pascal Frossard. Robustness via curvature regularization, and vice versa. *CVPR*, 2019b.

Edward Moroshko, Blake E Woodworth, Suriya Gunasekar, Jason D Lee, Nati Srebro, and Daniel Soudry. Implicit bias in deep linear classification: Initialization scale vs training accuracy. In *Advances in Neural Information Processing Systems*, volume 33, pages 22182–22193, 2020.

M. Mosbach, M. Andriushchenko, T. Trost, M. Hein, and D. Klakow. Logit pairing methods can fool gradient-based attacks. In *NeurIPS 2018 Workshop on Security in Machine Learning*, 2018.

Marius Mosbach, Maksym Andriushchenko, and Dietrich Klakow. On the stability of fine-tuning bert: Misconceptions, explanations, and strong baselines. In *ICLR*, 2021.

Rotem Mulayoff and Tomer Michaeli. Unique properties of flat minima in deep networks. In *ICML*, 2020.

Rotem Mulayoff, Tomer Michaeli, and Daniel Soudry. The implicit bias of minima stability: A view from function space. *Advances in Neural Information Processing Systems*, 34: 17749–17761, 2021.

Alan F Murray and Peter J Edwards. Synaptic weight noise during MLP learning enhances fault-tolerance, generalization and learning trajectory. In *NeurIPS*, 1993.

Aamir Mustafa, Salman Khan, Munawar Hayat, Roland Goecke, Jianbing Shen, and Ling Shao. Adversarial defense by restricting the hidden space of deep neural networks. *ICCV*, 2019.

Mor Shpigel Nacson, Kavya Ravichandran, Nathan Srebro, and Daniel Soudry. Implicit bias of the step size in linear diagonal neural networks. In *International Conference on Machine Learning*, pages 16270–16295. PMLR, 2022.

Vaishnavh Nagarajan and J Zico Kolter. Uniform convergence may be unable to explain generalization in deep learning. In *NeurIPS*, 2019.

Preetum Nakkiran. Learning rate annealing can provably help generalization, even for convex problems. *arXiv preprint arXiv:2005.07360*, 2020.

Preetum Nakkiran, Gal Kaplun, Dimitris Kalimeris, Tristan Yang, Benjamin L Edelman, Fred Zhang, and Boaz Barak. Sgd on neural networks learns functions of increasing complexity. *arXiv preprint arXiv:1905.11604*, 2019.

Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. Deep double descent: Where bigger models and more data hurt. In *ICLR*, 2020.

N. Narodytska and S. Kasiviswanathan. Simple black-box adversarial attacks on deep neural networks. In *CVPR Workshops*, 2017.

A. S. Nemirovsky and D. B. Yudin. *Problem Complexity and Method Efficiency in Optimization*. Wiley-Interscience Series in Discrete Mathematics. John Wiley & Sons, 1983.

Y. Nesterov and V. Spokoiny. Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics*, 17(2):527–566, 2017.

Yurii Nesterov. *Introductory Lectures on Convex Optimization*. Kluwer Academic, 2004.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. *Technical Report*, 2011.

Gergely Neu. Information-theoretic generalization bounds for stochastic gradient descent. In *COLT*, 2021.

Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. In search of the real inductive bias: On the role of implicit regularization in deep learning. In *ICLR workshops*, 2015.

Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nati Srebro. Exploring generalization in deep learning. In *NeurIPS*, 2017.

Maria-Irina Nicolae, Mathieu Sinn, Minh Ngoc Tran, Beat Buesser, Ambrish Rawat, Martin Wistuba, Valentina Zantedeschi, Nathalie Baracaldo, Bryant Chen, Heiko Ludwig, Ian Molloy, and Ben Edwards. Adversarial robustness toolbox v1.2.0. *arXiv preprint arXiv:1807.01069*, 2018.

Bernt Oksendal. *Stochastic differential equations: an introduction with applications*. Springer Science & Business Media, 2013.

OpenAI. Gpt-4 technical report. *arXiv: 2303.08774*, 2023.

Tianyu Pang, Kun Xu, Yinpeng Dong, Chao Du, Ning Chen, and Jun Zhu. Rethinking softmax cross-entropy loss for adversarial robustness. *ICLR*, 2020a.

Tianyu Pang, Kun Xu, and Jun Zhu. Mixup inference: Better exploiting mixup to defend adversarial attacks. In *ICLR*, 2020b.

Tianyu Pang, Xiao Yang, Yinpeng Dong, Kun Xu, Hang Su, and Jun Zhu. Boosting adversarial training with hypersphere embedding. *NeurIPS*, 2020c.

Tianyu Pang, Xiao Yang, Yinpeng Dong, Hang Su, and Jun Zhu. Bag of tricks for adversarial training. *ICLR*, 2021a.

Tianyu Pang, Huishuai Zhang, Di He, Yinpeng Dong, Hang Su, Wei Chen, Jun Zhu, and Tie-Yan Liu. Adversarial training with rectified rejection. *arXiv preprint arXiv:2105.14785*, 2021b.

N. Papernot, P. McDaniel, and I. Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. arXiv preprint arXiv:1605.07277, 2016a.

N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami. Distillation as a defense to adversarial perturbations against deep networks. In *IEEE Symposium on Security & Privacy*, 2016b.

Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. *ASIA CCS'17*, 2017.

Nicolas Papernot, Fartash Faghri, Nicholas Carlini, Ian Goodfellow, Reuben Feinman, Alexey Kurakin, Cihang Xie, Yash Sharma, Tom Brown, Aurko Roy, Alexander Matyasko, Vahid Behzadan, Karen Hambardzumyan, Zhishuai Zhang, Yi-Lin Juang, Zhi Li, Ryan Sheatsley, Abhibhav Garg, Jonathan Uesato, Willi Gierke, Yinpeng Dong, David Berthelot, Paul Hendricks, Jonas Rauber, and Rujun Long. Technical report on the cleverhans v2.1.0 adversarial examples library. *arXiv preprint arXiv:1610.00768*, 2018.

Namuk Park and Songkuk Kim. How do vision transformers work? *ICLR*, 2022.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. *Technical Report*, 2017.

Scott Pesme, Loucas Pillaud-Vivien, and Nicolas Flammarion. Implicit bias of sgd for diagonal linear networks: a provable benefit of stochasticity. In *NeurIPS*, 2021.

Henning Petzka, Michael Kamp, Linara Adilova, Cristian Sminchisescu, and Mario Boley. Relative flatness and generalization. *NeurIPS*, 34:18420–18432, 2021.

Loucas Pillaud-Vivien, Julien Reygner, and Nicolas Flammarion. Label noise (stochastic) gradient descent implicitly solves the lasso for quadratic parametrisation. In *Conference on Learning Theory*, 2022.

Chongli Qin, James Martens, Sven Gowal, Dilip Krishnan, Krishnamurthy Dvijotham, Alhussein Fawzi, Soham De, Robert Stanforth, and Pushmeet Kohli. Adversarial robustness through local linearization. *NeurIPS*, 2019.

Rahul Rade and Seyed-Mohsen Moosavi-Dezfooli. Helper-based adversarial training: Reducing excessive margin to achieve a better accuracy vs. robustness trade-off. *OpenReview*, 2021.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021.

Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified defenses against adversarial examples. *ICLR*, 2018.

Aditi Raghunathan, Sang Michael Xie, Fanny Yang, John Duchi, and Percy Liang. Understanding and mitigating the tradeoff between robustness and accuracy. In *ICML*, 2020.

Mohammad Saidur Rahman, Mohsen Imani, Nate Mathews, and Matthew Wright. Mockingbird: Defending against deep-learning-based website fingerprinting attacks with ad-

versarial traces. *IEEE Transactions on Information Forensics and Security*, 16:1594–1609, 2020.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with CLIP latents. *arXiv preprint arXiv:2204.06125*, 2022.

L. Rastrigin. The convergence of the random search method in the extremal control of a many parameter system. *Automaton & Remote Control*, 24:1337–1342, 1963.

J. Rauber, W. Brendel, and M. Bethge. Foolbox: A python toolbox to benchmark the robustness of machine learning models. In *ICML Reliable Machine Learning in the Wild Workshop*, 2017.

Sylvestre-Alvise Rebuffi, Sven Gowal, Dan Andrei Calian, Florian Stimberg, Olivia Wiles, and Timothy A Mann. Data augmentation can improve robustness. In *NeurIPS*, 2021.

Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? In *ICML*, 2019.

Leslie Rice, Eric Wong, and J. Zico Kolter. Overfitting in adversarially robust deep learning. *ICML*, 2020.

Jérôme Rony, Eric Granger, Marco Pedersoli, and Ismail Ben Ayed. Augmented lagrangian adversarial attacks. *arXiv preprint arXiv:2011.11857*, 2020.

Jérôme Rony, Luiz G. Hafemann, Luiz S. Oliveira, Ismail Ben Ayed, Robert Sabourin, and Eric Granger. Decoupling direction and norm for efficient gradient-based l2 adversarial attacks and defenses. *CVPR*, 2019.

Andrew Slavin Ross and Finale Doshi-Velez. Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients. *AAAI*, 2018.

Parsa Saadatpanah, Ali Shafahi, and Tom Goldstein. Adversarial attacks on copyright detection systems. In *ICML*, 2020.

Hadi Salman, Andrew Ilyas, Logan Engstrom, Ashish Kapoor, and Aleksander Madry. Do adversarially robust imagenet models transfer better? *NeurIPS*, 2020.

Pouya Samangouei, Maya Kabkab, and Rama Chellappa. Defense-GAN: Protecting classifiers against adversarial attacks using generative models. In *ICLR*, 2018.

Shibani Santurkar, Dimitris Tsipras, Brandon Tran, Andrew Ilyas, Logan Engstrom, and Aleksander Madry. Image synthesis with a single (robust) classifier. *NeurIPS*, 2019.

Christian Schlarmann and Matthias Hein. On the adversarial robustness of multi-modal foundation models. *arXiv preprint arXiv:2308.10741*, 2023.

Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

G. Schrack and M. Choit. Optimized relative step size random searches. *Mathematical Programming*, 10:230–244, 1976.

M. Schumer and K. Steiglitz. Adaptive step size random search. *IEEE Transactions on Automatic Control*, 13(3):270–276, 1968.

Leo Schwinn, René Raab, An Nguyen, Dario Zanca, and Bjoern Eskofier. Exploring robust misclassifications of neural networks to enhance adversarial attacks. *arXiv preprint arXiv:2105.10304*, 2021.

Vikash Sehwag, Arjun Nitin Bhagoji, Liwei Song, Chawin Sitawarin, Daniel Cullina, Mung Chiang, and Prateek Mittal. Analyzing the robustness of open-world machine learning. In *12th ACM Workshop on Artificial Intelligence and Security*, 2019.

Vikash Sehwag, Shiqi Wang, Prateek Mittal, and Suman Jana. Hydra: Pruning adversarially robust neural networks. *NeurIPS*, 2020a.

Vikash Sehwag, Shiqi Wang, Prateek Mittal, and Suman Jana. On pruning adversarially robust neural networks. *NeurIPS*, 2020b.

Vikash Sehwag, Saeed Mahloujifar, Tinashe Handina, Sihui Dai, Chong Xiang, Mung Chiang, and Prateek Mittal. Improving adversarial robustness using proxy distributions. *arXiv*, 2021.

M. Seungyong, A. Gaon, and O. S. Hyun. Parsimonious black-box adversarial attacks via efficient combinatorial optimization. In *ICML*, 2019.

Ali Shafahi, Mahyar Najibi, Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S. Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free! *NeurIPS*, 2019.

Uri Shaham, Yutaro Yamada, and Sahand Negahban. Understanding adversarial training: Increasing local stability of supervised models through robust optimization. *Neurocomputing*, 2018.

Shawn Shan, Emily Wenger, Jiayun Zhang, Huiying Li, Haitao Zheng, and Ben Y Zhao. Fawkes: Protecting privacy against unauthorized deep learning models. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 1589–1604, 2020.

Changhao Shi, Chester Holtz, and Gal Mishne. Online adversarial purification based on self-supervised learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=_i3ASPp12WS.

Sungbin Shin, Dongyeop Lee, Maksym Andriushchenko, and Namhoon Lee. The effects of overparameterization on sharpness-aware minimization: An empirical and theoretical analysis. *ICML 2023 Workshop on High-dimensional Learning Dynamics*, 2023.

Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18. IEEE, 2017.

S. N. Shukla, A. K. Sahu, D. Willmott, and Z. Kolter. Black-box adversarial attacks with Bayesian optimization. arXiv preprint arXiv:1909.13857, 2019.

Carl-Johann Simon-Gabriel, Yann Ollivier, Leon Bottou, Bernhard Schölkopf, and David Lopez-Paz. First-order adversarial vulnerability of neural networks and input dimension. *ICML*, 2019.

Mayank Singh, Abhishek Sinha, Nupur Kumari, Harshitha Machiraju, Balaji Krishnamurthy, and Vineeth N Balasubramanian. Harnessing the vulnerability of latent layers in adversarially trained models. *IJCAI*, 2019.

Chawin Sitawarin, Supriyo Chakraborty, and David Wagner. Improving adversarial robustness through progressive hardening. *arXiv*, 2020.

Leslie N Smith. Cyclical learning rates for training neural networks. *WACV*, 2017.

Samuel L. Smith and Quoc V. Le. A Bayesian perspective on generalization and stochastic gradient descent. In *International Conference on Learning Representations*, 2018.

Samuel L Smith, Benoit Dherin, David GT Barrett, and Soham De. On the origin of implicit regularization in stochastic gradient descent. *arXiv preprint arXiv:2101.12176*, 2021.

Liwei Song and Prateek Mittal. Systematic evaluation of privacy risks of machine learning models. In *30th USENIX Security Symposium (USENIX Security 21)*, 2021.

Liwei Song, Reza Shokri, and Prateek Mittal. Privacy risks of securing machine learning models against adversarial examples. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 241–257, 2019.

Liwei Song, Vikash Sehwag, Arjun Nitin Bhagoji, and Prateek Mittal. A critical evaluation of open-world machine learning. *arXiv preprint arXiv:2007.04391*, 2020.

Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The implicit bias of gradient descent on separable data. *The Journal of Machine Learning Research*, 19(1):2822–2878, 2018.

Kaustubh Sridhar, Oleg Sokolsky, Insup Lee, and James Weimer. Robust learning via persistency of excitation. *arXiv*, 2021.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, 2014.

Andreas Steiner, Alexander Kolesnikov, Xiaohua Zhai, Ross Wightman, Jakob Uszkoreit, and Lucas Beyer. How to train your vit? data, augmentation, and regularization in vision transformers. *TMLR*, 2021.

David Stutz, Matthias Hein, and Bernt Schiele. Relating adversarially robust generalization to flat minima. In *ICCV*, 2021.

J. Su, D. V. Vargas, and K. Sakurai. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 2019.

Fnu Suya, Jianfeng Chi, David Evans, and Yuan Tian. Hybrid batch attacks: Finding black-box adversarial examples with limited queries. arXiv preprint, arXiv:1908.07000, 2019.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *ICLR*, 2014.

Lue Tao, Lei Feng, Jinfeng Yi, Sheng-Jun Huang, and Songcan Chen. Provable defense against delusive poisoning. *arXiv preprint arXiv:2102.04716*, 2021.

Rohan Taori, Achal Dave, Vaishaal Shankar, Nicholas Carlini, Benjamin Recht, and Ludwig Schmidt. Measuring robustness to natural distribution shifts in image classification. *arXiv preprint arXiv:2007.00644*, 2020a.

Rohan Taori, Achal Dave, Vaishaal Shankar, Nicholas Carlini, Benjamin Recht, and Ludwig Schmidt. Measuring robustness to natural distribution shifts in image classification. *NeurIPS*, 2020b.

V. Tjeng, K. Xiao, and R. Tedrake. Evaluating robustness of neural networks with mixed integer programming. In *ICLR*, 2019a.

Vincent Tjeng, Kai Xiao, and Russ Tedrake. Evaluating robustness of neural networks with mixed integer programming. *ICLR*, 2019b.

Giorgos Tolias, Filip Radenovic, and Ondrej Chum. Targeted mismatch adversarial attack: Query with a flower to retrieve the tower. In *ICCV*, 2019.

Florian Tramèr and Dan Boneh. Adversarial training and robustness for multiple perturbations. In *NeurIPS*, 2019.

Florian Tramèr, Pascal Dupré, Gili Rusak, Giancarlo Pellegrino, and Dan Boneh. Adversarial: Perceptual ad blocking meets adversarial machine learning. In *ACM SIGSAC CCS*, 2019.

Florian Tramèr, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. On adaptive attacks to adversarial example defenses. In *NeurIPS*, 2020.

F. Tramèr and D. Boneh. Adversarial training and robustness for multiple perturbations. In *NeurIPS*, 2019.

Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. *ICLR*, 2018.

Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. *ICLR*, 2019.

Yusuke Tsuzuku, Issei Sato, and Masashi Sugiyama. Normalized flat minima: Exploring scale invariant definition of flat minima for neural networks using pac-bayesian analysis. In *International Conference on Machine Learning*, pages 9636–9647. PMLR, 2020.

C.-C. Tu, P. Ting, P.-Y. Chen, S. Liu, H. Zhang, J. Yi, C.-J. Hsieh, and S.-M. Cheng. Autozoom: Autoencoder-based zeroth order optimization method for attacking black-box neural networks. In *AAAI Conference on Artificial Intelligence*, 2019.

J. Uesato, B. O'Donoghue, A. Van den Oord, and P. Kohli. Adversarial risk and the dangers of evaluating against weak attacks. In *ICML*, 2018.

Jonathan Uesato, Jean-Baptiste Alayrac, Po-Sen Huang, Robert Stanforth, Alhussein Fawzi, and Pushmeet Kohli. Are labels required for improving adversarial robustness? *NeurIPS*, 2019.

Francisco Utrera, Evan Kravitz, N Benjamin Erichson, Rajiv Khanna, and Michael W Mahoney. Adversarially-trained deep nets transfer better. *arXiv preprint arXiv:2007.05869*, 2020.

Tomas Vaskevicius, Varun Kanade, and Patrick Rebeschini. Implicit regularization for optimal sparse recovery. *Advances in Neural Information Processing Systems*, 32, 2019.

Shanmukha Ramakrishna Vedantam, David Lopez-Paz, and David J. Schwab. An empirical investigation of domain generalization with empirical risk minimizers. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *NeurIPS*, 2021.

B. S. Vivek and R. Venkatesh Babu. Single-step adversarial training with dropout scheduling. *CVPR*, 2020.

Dequan Wang, An Ju, Evan Shelhamer, David Wagner, and Trevor Darrell. Fighting gradients with gradients: Dynamic defenses against adversarial attacks. *arXiv preprint arXiv:2105.08714*, 2021.

Haohan Wang, Songwei Ge, Zachary Lipton, and Eric P Xing. Learning robust global representations by penalizing local predictive power. In *NeurIPS*, pages 10506–10518, 2019a.

Jianyu Wang and Haichao Zhang. Bilateral adversarial training: Towards fast training of more robust models against adversarial attacks. *ICCV*, 2019.

Yisen Wang, Xingjun Ma, James Bailey, Jinfeng Yi, Bowen Zhou, and Quanquan Gu. On the convergence and robustness of adversarial training. *ICML*, 2019b.

Yisen Wang, Difan Zou, Jinfeng Yi, James Bailey, Xingjun Ma, and Quanquan Gu. Improving adversarial robustness requires revisiting misclassified examples. *ICLR*, 2020.

Yuqing Wang, Minshuo Chen, Tuo Zhao, and Molei Tao. Large learning rate tames homogeneity: Convergence and balancing effect. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=3tbDrs77LJ5.

Ziqiao Wang and Yongyi Mao. On the generalization of models trained with SGD: Information-theoretic bounds and implications. In *ICLR*, 2022.

Michael L. Waskom. Seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60):3021, 2021. doi: 10.21105/joss.03021. URL https://doi.org/10.21105/joss.03021.

Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. Jailbroken: How does llm safety training fail? In *NeurIPS*, 2023.

Wei Wen, Yandan Wang, Feng Yan, Cong Xu, Chunpeng Wu, Yiran Chen, and Hai Li. Smoothout: Smoothing out sharp minima to improve generalization in deep learning. *arXiv preprint arXiv:1805.07898*, 2018.

Tsui-Wei Weng, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Duane Boning, Inderjit S. Dhillon, and Luca Daniel. Towards fast computation of certified robustness for relu networks. *ICML*, 2018.

Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *NAACL*, 2018.

Stephan Wojtowytsch. Stochastic gradient descent with noise of machine learning type. Part II: Continuous time analysis. *arXiv preprint arXiv:2106.02588*, 2021a.

Stephan Wojtowytsch. Stochastic gradient descent with noise of machine learning type. Part I: Discrete time analysis. *arXiv preprint arXiv:2105.01650*, 2021b.

Eric Wong and J Zico Kolter. Learning perturbation sets for robust machine learning. *arXiv preprint arXiv:2007.08450*, 2020.

Eric Wong and Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. *ICML*, 2018.

Eric Wong, Frank R Schmidt, and J Zico Kolter. Wasserstein adversarial examples via projected sinkhorn iterations. In *ICML*, 2019.

Eric Wong, Leslie Rice, and J. Zico Kolter. Fast is better than free: Revisiting adversarial training. *ICLR*, 2020.

Blake Woodworth, Suriya Gunasekar, Jason D. Lee, Edward Moroshko, Pedro Savarese, Itay Golan, Daniel Soudry, and Nathan Srebro. Kernel and rich regimes in overparametrized models. In *COLT*, 2020.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *ICML*, pages 23965–23998. PMLR, 2022a.

Mitchell Wortsman, Gabriel Ilharco, Jong Wook Kim, Mike Li, Simon Kornblith, Rebecca Roelofs, Raphael Gontijo Lopes, Hannaneh Hajishirzi, Ali Farhadi, Hongseok Namkoong, et al. Robust fine-tuning of zero-shot models. In *CVPR*, pages 7959–7971, 2022b.

Boxi Wu, Jinghui Chen, Deng Cai, Xiaofei He, and Quanquan Gu. Do wider neural networks really help adversarial robustness? *arXiv*, 2020a.

Dongxian Wu, Shu tao Xia, and Yisen Wang. Adversarial weight perturbation helps robust generalization. *NeurIPS*, 2020b.

Jingfeng Wu, Difan Zou, Vladimir Braverman, and Quanquan Gu. Direction matters: On the implicit bias of stochastic gradient descent with moderate learning rate. *International Conference on Learning Representations*, 2021.

Lei Wu, Chao Ma, et al. How sgd selects the global minima in over-parameterized learning: A dynamical stability perspective. *Advances in Neural Information Processing Systems*, 31, 2018.

Lei Wu, Mingze Wang, and Weijie Su. When does sgd favor flat minima? a quantitative characterization via linear stability. *NeurIPS*, 2022.

Yuxin Wu and Kaiming He. Group normalization. In *ECCV*, 2018.

Chang Xiao, Peilin Zhong, and Changxi Zheng. Enhancing adversarial defense by k-winners-take-all. *ICLR*, 2020.

Cihang Xie, Jianyu Wang, Zhishuai Zhang, Yuyin Zhou, Lingxi Xie, and Alan Yuille. Adversarial examples for semantic segmentation and object detection. In *ICCV*, 2017.

Cihang Xie, Mingxing Tan, Boqing Gong, Jiang Wang, Alan L Yuille, and Quoc V Le. Adversarial examples improve image recognition. In *CVPR*, 2020.

Chen Xing, Devansh Arpit, Christos Tsirigotis, and Yoshua Bengio. A walk with sgd. *arXiv preprint arXiv:1802.08770*, 2018.

Cong Xu, Xiang Li, and Min Yang. An orthogonal classifier for improving the adversarial robustness of neural networks. *arXiv preprint arXiv:2105.09109*, 2021.

Han Xu, Xiaorui Liu, Yaxin Li, and Jiliang Tang. To be robust or to be fair: Towards fairness in adversarial training. *arXiv preprint arXiv:2010.06121*, 2020.

Z. Yan, Y. Guo, and C. Zhang. Subspace attack: Exploiting promising subspaces for query-efficient black-box attacks. In *NeurIPS*, 2019.

Ning Yang, Chao Tang, and Yuhai Tu. Stochastic gradient descent introduces an effective landscape-dependent regularization favoring flat solutions. *arXiv preprint arXiv:2206.01246*, 2022.

Yao-Yuan Yang, Cyrus Rashtchian, Hongyang Zhang, Ruslan Salakhutdinov, and Kamalika Chaudhuri. A closer look at accuracy vs. robustness. *Advances in Neural Information Processing Systems*, 33, 2020.

Yuzhe Yang, Guo Zhang, Dina Katabi, and Zhi Xu. Me-net: Towards effective adversarial robustness with matrix estimation. In *ICML*, 2019.

Zhengyuan Yang, Linjie Li, Kevin Lin, Jianfeng Wang, Chung-Ching Lin, Zicheng Liu, and Lijuan Wang. The dawn of lmms: Preliminary explorations with gpt-4v (ision). *arXiv preprint arXiv:2309.17421*, 2023.

D Yin, R. G. Lopes, J. Shlens, E. D Cubuk, and J. Gilmer. A Fourier perspective on model robustness in computer vision. In *NeurIPS*, 2019.

Fisher Yu, Vladlen Koltun, and Thomas Funkhouser. Dilated residual networks. In *CVPR*, 2017.

Yaodong Yu, Zitong Yang, Edgar Dobriban, Jacob Steinhardt, and Yi Ma. Understanding generalization in adversarial training via the bias-variance decomposition. *arXiv preprint arXiv:2103.09947*, 2021.

Zelda B Zabinsky. Random search algorithms. *Wiley encyclopedia of operations research and management science*, 2010.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, 2016.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *ICLR*, 2017a.

Dinghuai Zhang, Tianyuan Zhang, Yiping Lu, Zhanxing Zhu, and Bin Dong. You only propagate once: Accelerating adversarial training via maximal principle. *NeurIPS*, 2019a.

Guodong Zhang, Chaoqi Wang, Bowen Xu, and Roger Grosse. Three mechanisms of weight decay regularization. In *International Conference on Learning Representations*, 2018.

H. Zhang, Y. Yu, J. Jiao, E. P. Xing, L. El Ghaoui, and M. I. Jordan. Theoretically principled trade-off between robustness and accuracy. In *ICML*, 2019b.

Haichao Zhang and Jianyu Wang. Defense against adversarial attacks using feature scattering-based adversarial training. *NeurIPS*, 2019.

Haichao Zhang and Wei Xu. Adversarial interpolation training: A simple approach for improving model robustness. *OpenReview*, 2019.

Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P. Xing, Laurent El Ghaoui, and Michael I. Jordan. Theoretically principled trade-off between robustness and accuracy. In *ICML*, 2019c.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017b.

Huan Zhang, Hongge Chen, Chaowei Xiao, Sven Gowal, Robert Stanforth, Bo Li, Duane Boning, and Cho-Jui Hsieh. Towards stable and efficient training of verifiably robust neural networks. *ICLR*, 2020a.

Jingfeng Zhang, Xilie Xu, Bo Han, Gang Niu, Lizhen Cui, Masashi Sugiyama, and Mohan Kankanhalli. Attacks which do not kill training make adversarial learning stronger. *ICML*, 2020b.

Jingfeng Zhang, Jianing Zhu, Gang Niu, Bo Han, Masashi Sugiyama, and Mohan Kankanhalli. Geometry-aware instance-reweighted adversarial training. *ICLR*, 2021a.

Shuofeng Zhang, Isaac Reid, Guillermo Valle Pérez, and Ard Louis. Why flatness does and does not correlate with generalization for deep neural networks. *arXiv preprint arXiv:2103.06219*, 2021b.

Xiao Zhang and David Evans. Incorporating label uncertainty in understanding adversarial robustness. *arXiv preprint arXiv:2107.03250*, 2021.

Hao Zhao, Maksym Andriushchenko, Francesco Croce, and Nicolas Flammarion. Long is more for alignment: A simple but tough-to-beat baseline for instruction fine-tuning. *ICLR 2024 Data-Centric Machine Learning Research Workshop*, 2024.

S. Zheng, Y. Song, T. Leung, and I. J. Goodfellow. Improving the robustness of deep neural networks via stability training. In *CVPR*, 2016.

T. Zheng, C. Chen, and K. Ren. Distributionally adversarial attack. In *AAAI*, 2019.

Yaowei Zheng, Richong Zhang, and Yongyi Mao. Regularizing neural networks via adversarial model perturbation. In *CVPR*, 2021.

Jiachen Zhong, Xuanqing Liu, and Cho-Jui Hsieh. Improving the speed and quality of gan by adversarial training. *arXiv preprint arXiv:2008.03364*, 2020.

Pan Zhou, Jiashi Feng, Chao Ma, Caiming Xiong, Steven Chu Hong Hoi, et al. Towards theoretically understanding why SGD generalizes better than Adam in deep learning. *NeurIPS*, 2020.

Chen Zhu, Yu Cheng, Zhe Gan, Siqi Sun, Tom Goldstein, and Jingjing Liu. Freelb: Enhanced adversarial training for natural language understanding. *ICLR*, 2019.

Juntang Zhuang, Boqing Gong, Liangzhe Yuan, Yin Cui, Hartwig Adam, Nicha C Dvornek, sekhar tatikonda, James s Duncan, and Ting Liu. Surrogate gap minimization improves sharpness-aware training. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=edONMAnhLu-.

Liu Ziyin, Kangqiao Liu, Takashi Mori, and Masahito Ueda. Strength of minibatch noise in sgd. In *International Conference on Learning Representations*, 2022.

Andy Zou, Zifan Wang, J Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*, 2023.

Maksym Andriushchenko – Curriculum Vitae

PERSONAL DATA

Site: https://andriushchenko.me/
Email: maksym@andriushchenko.me
Scholar: https://scholar.google.com/citations?user=ZNtuJYoAAAAJ
Github: https://github.com/max-andr/

EDUCATION

École Polytechnique Fédérale de Lausanne (EPFL), Switzerland *(Sep 2019 - now)*
PhD student in Computer Science advised by Nicolas Flammarion

Saarland University, Germany *(Oct 2016 – Aug 2019)*
Master's Degree in Computer Science advised by Matthias Hein from the University of Tübingen

Dnipro National University of Railway Transport, Ukraine (*Sep 2012 – June 2016)*
Bachelor's Degree in Software Engineering — *with honors*

EMPLOYMENT

Adobe Research, Media Intelligence Lab, remote	**Time**: July 2021 – October 2021 **Role**: Research internship supervised by John Collomosse
PrivatBank, Dnipro, Ukraine	**Time:** November 2015 – June 2016 **Role**: Part-time data scientist (predictive modeling, e-commerce personalization)

AWARDS

Grant and scholarship awards	**Google PhD fellowship 2022-2025 ($80k per year for 3 years)** **Open Philanthropy AI PhD Fellowship 2022-2024 ($10k per year for travel/equipment)** Google Research Collab 2022-2023 ($80k for one year + $20k in cloud compute) EDIC PhD fellowship from EPFL for the first year ($60k) DAAD MSc scholarship for 2 years to study at Saarland University ($20k)
Awards for papers and competitions	Swiss AI Safety Prize (2024): **award for one of the top paper submissions** Joint Conference of Korean AI Association (2023): **best paper award** ICLR Workshop on Security & Safety in ML Systems (2021): **best paper honorable mention** Swiss Machine Learning Day (2019): **best paper award**

ACADEMIC SERVICE

Participant	Red teaming of the **OpenAI fine-tuning** service as an external expert (October 2023) Robust AI 4-day workshop organized by AirBus AI Research and TNO (January 2021)
Reviewer	NeurIPS'23, ICML'23, NeurIPS'22 (**top reviewer**), ICML'22, NeurIPS'21, ICML'21, CVPR'21, ICLR'21 (**outstanding reviewer**), NeurIPS'20 (**top 10% reviewers**)
Program committee in workshops	**NeurIPS'23** "R0-FoMo Workshop on Robustness of Few-shot and Zero-shot Learning in Foundation Models", **NeurIPS'23** "Workshop on Distribution Shifts: New Frontiers with Foundation Models", **ICML'23** "2nd ICML Workshop on New Frontiers in Adversarial ML", **ICLR'23** "Workshop on Pitfalls of Limited Data and Computation for Trustworthy ML", **NeurIPS'22** "Workshop on Distribution Shifts", **NeurIPS'22** "ML Safety Workshop", **ICML'22** "New Frontiers in Adversarial Machine Learning", **ICML'22** "Principles of Distribution Shift", **NeurIPS'21**: "Distribution Shifts: Connecting Methods and Applications", **ICML'21** "Uncertainty and Robustness in Deep Learning", **CVPR'21** "Adversarial ML in Real-World Computer Vision Systems", **ICLR'21** "Robust and Reliable ML in the Real World", "Security and Safety in ML Systems", **ICML'20** "Uncertainty and Robustness in Deep Learning", **CVPR'20** "Adversarial ML in Computer Vision", **ICLR'20** "Towards Trustworthy ML" (**best reviewer award**)
Outreach activities	National coordinator for Switzerland at #ScienceForUkraine Coordinator for Switzerland and admission officer at the Ukrainian Global University AI and STEM workshop at a summer camp for displaced Ukrainian children in Romania

STUDENT SUPERVISION

Hao Zhao	**MSc thesis (2023)**: "Long Is More for Alignment: A Simple but Tough-to-Beat Baseline for Instruction Fine-Tuning"
Hichem Hadhri	**MSc project (2023)**: "Understanding overfitting in large language models"
Tiberiu Musat	**BSc project (2023)**: "Investigating key components for fast optimization of deep networks"
Francesco d'Angelo	**PhD semester project (2023):** "Understanding the role of weight decay in deep learning"
Théau Vannier	**MSc project (2023):** "Understanding the training instability of transformers"
Joshua Freeman	**BSc project (2022, unofficial)**: "Automatic recognition of unexploded ordnance using transfer learning"
Jana Vuckovic	**MSc project (2022)**: "Rethinking the relationship between sharpness and generalization" (follow-up work is published at ICML'23)
Mehrdad Saberi	**Summer internship (2021):** "Wasserstein adversarial training and perceptual robustness"
Edoardo Debenedetti	**MSc project (2021)**: "RobustBench: a standardized adversarial robustness benchmark" (published at NeurIPS'21 Datasets and Benchmarks Track)
Klim Kireev	**PhD semester project (2020)**: "On the effectiveness of adversarial training against common corruptions" (published at UAI'22)
Etienne Bonvin	**MSc project (2020)**: "Adversarial robustness of kernel methods"
Oriol Barbany	**MSc project (2019)**: "Affine-invariant robust training"

TEACHING EXPERIENCE

EPFL	**Probability & Statistics 2021, 2022** (by E. Abbé), **Machine Learning 2020, 2021, 2022, 2023** (by M. Jaggi, N. Flammarion), **Advanced Algorithms 2020** (by M. Kapralov)
MPI for Informatics	**Machine Learning 2018-2019** (lecturer: B. Schiele)
Saarland University	**Neural Networks: Implementation and Application 2017** (lecturer: D. Klakow)

SELECTED PUBLICATIONS

H. Zhao, **M. Andriushchenko**, F. Croce, N. Flammarion. Long Is More for Alignment: A Simple but Tough-to-Beat Baseline for Instruction Fine-Tuning (ICLR 2024 Data-Centric Machine Learning Research Workshop) [paper]

M. Andriushchenko. Adversarial Attacks on GPT-4 via Simple Random Search (December 2023, **one of the top submissions for** Swiss AI Safety Prize) [paper]

M. Andriushchenko, F. Croce, M. Müller, M. Hein, N. Flammarion. A Modern Look at the Relationship between Sharpness and Generalization (ICML 2023) [paper]

M. Andriushchenko, A. Varre, L. Pillaud-Vivien, N. Flammarion. SGD with Large Step Sizes Learns Sparse Features (ICML 2023) [paper]

F. Croce*, **M. Andriushchenko***, V. Sehwag*, E. Debenedetti*, N. Flammarion, M. Chiang, P. Mittal, M. Hein. RobustBench: a standardized adversarial robustness benchmark (NeurIPS 2021 Datasets and Benchmarks Track, **Best Paper Honorable Mention Prize** at ICLR 2021 Workshop on Security and Safety in Machine Learning Systems) [paper]

M. Andriushchenko, N. Flammarion. Understanding and Improving Fast Adversarial Training (NeurIPS 2020) [paper]

M. Andriushchenko*, F. Croce*, N. Flammarion, M. Hein. Square Attack: a Query-Efficient Black-Box Adversarial Attack via Random Search (ECCV 2020) [paper]

FULL PUBLICATION LIST

H. Zhao, **M. Andriushchenko**, F. Croce, N. Flammarion. Long Is More for Alignment: A Simple but Tough-to-Beat Baseline for Instruction Fine-Tuning (ICLR 2024 Data-Centric Machine Learning Research Workshop) [paper]

M. Andriushchenko. Adversarial Attacks on GPT-4 via Simple Random Search (December 2023, **one of the top submissions for** Swiss AI Safety Prize) [paper]

E. Debenedetti, Z. Wan, **M. Andriushchenko**, V. Sehwag, K. Bhardwaj, B. Kailkhura. Scaling Compute Is Not All You Need for Adversarial Robustness (ICLR 2024 Workshop on Reliable and Responsible Foundation Models) [paper]

M. Andriushchenko*, F. D'Angelo*, A. Varre, N. Flammarion. Why Do We Need Weight Decay in Modern Deep Learning? (NeurIPS 2023 Workshop on Mathematics of Modern Machine Learning) [paper]

L. Adilova, **M. Andriushchenko**, M. Kamp, A. Fischer, M. Jaggi. Layer-Wise Linear Mode Connectivity (ICLR 2024) [paper]

S. Shin, D. Lee, **M. Andriushchenko**, N. Lee. The Effects of Overparameterization on Sharpness-Aware Minimization: An Empirical and Theoretical Analysis (September 2023, **best paper award** at the Joint Conference of Korean Artificial Intelligence Association (2023) [paper])

M. Andriushchenko, D. Bahri, H. Mobahi, N. Flammarion. Sharpness-Aware Minimization Leads to Low-Rank Features (NeurIPS 2023) [paper]

K. Kireev, **M. Andriushchenko**, C. Troncoso, N. Flammarion. Transferable Adversarial Robustness for Categorical Data via Universal Robust Embeddings (NeurIPS 2023) [paper]

M. Andriushchenko, F. Croce, M. Müller, M. Hein, N. Flammarion. A modern look at the relationship between sharpness and generalization. (ICML 2023) [paper]

M. Andriushchenko, A. Varre, L. Pillaud-Vivien, N. Flammarion. SGD with large step sizes learns sparse features (ICML 2023) [paper]

K. Kireev*, **M. Andriushchenko***, N. Flammarion. On the effectiveness of adversarial training against common corruptions (UAI 2022, ICLR'21 Workshop on Robust and Reliable Machine Learning in the Real World) [paper]

Michael Rose, Sanita Reinsone, **Maksym Andriushchenko**, Marcin Bartosiak, Anna Bobak et al. #ScienceForUkraine: an Initiative to Support the Ukrainian Academic Community. "3 Months Since Russia's Invasion in Ukraine", February 26 – May 31, 2022 (SSRN, 2022) [paper]

M. Andriushchenko, N. Flammarion. Towards Understanding Sharpness-Aware Minimization (ICML 2022) [paper]

M. Andriushchenko, X. Rebecca Li, Geoffrey Oxholm, Thomas Gittings, Tu Bui, Nicolas Flammarion, John Collomosse. ARIA: Adversarially Robust Image Attribution for Content Provenance (CVPR 2022 Workshop on Media Forensics) [paper]

F. Croce, **M. Andriushchenko**, N. Singh, N. Flammarion, M. Hein. Sparse-RS: a versatile framework for query-efficient sparse black-box adversarial attacks (AAAI 2022) [paper]

F. Croce*, **M. Andriushchenko***, V. Sehwag*, E. Debenedetti*, N. Flammarion, M. Chiang, P. Mittal, M. Hein. RobustBench: a standardized adversarial robustness benchmark (NeurIPS 2021 Datasets and Benchmarks Track, **Best Paper Honorable Mention Prize** at ICLR 2021 Workshop on Security and Safety in Machine Learning Systems) [paper]

M. Mosbach, **M. Andriushchenko**, D. Klakow. On the Stability of Fine-tuning BERT: Misconceptions, Explanations, and Strong Baselines (ICLR 2021) [paper]

M. Andriushchenko*, F. Croce*, N. Flammarion, M. Hein. Square Attack: a query-efficient black-box adversarial attack via random search (ECCV 2020) [paper]

M. Andriushchenko, N. Flammarion. Understanding and Improving Fast Adversarial Training (NeurIPS 2020) [paper]

M. Andriushchenko, M. Hein. Provably Robust Boosted Decision Stumps and Trees against Adversarial Attacks (NeurIPS 2019, contributed talk at Workshop on Machine Learning with Guarantees; **best paper award** at Swiss Machine Learning Day (2019)) [paper]

M. Hein, **M. Andriushchenko**, J. Bitterwolf. Why ReLU networks yield high-confidence predictions far away from the training data and how to mitigate the problem (oral at CVPR 2019, 5.6% acceptance rate, contributed talk at ICML 2019 Uncertainty and Robustness in Deep Learning Workshop) [paper]

F. Croce*, **M. Andriushchenko***, M. Hein. Provable Robustness of ReLU Networks via Maximization of Linear Regions (AISTATS 2019) [paper]

M. Mosbach*, **M. Andriushchenko***, T. Trost, M. Hein, D. Klakow. Logit Pairing Methods Can Fool Gradient-Based Attacks (NeurIPS 2018 Workshop on Security in ML) [paper]

M. Hein and **M. Andriushchenko**. Formal Guarantees on the Robustness of a Classifier Against Adversarial Manipulation (NeurIPS 2017) [paper]